A PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SOLO PROGRAMADORES

) III, NÚMERO 39

975 Ptas.

¿QUÉ ES EL UML?

ANÁLISIS Y DISEÑO OO

CONTENIDO DEL CD-ROM

- Internet Explorer 4.0 w95 & NT4
- DirectX 5.0 & SDK
- Compilador DJGPP
- JDK 1.1.3 & JRE 1.1.3 para Linux
- Librerías OpenGL
- GetRight 2.11 & VideoPhone 2.5
- Linux Howtos & Mini-Howtos actualizados
- Y Mucho más

SISTEMAS ABIERTOS
Creamos una intranet con LINUX

PROGRAMAS DE AJEDREZ
Movimiento selectivo. Extensores

WWW
Netscape Netcaster y JavaScript

LINUX
Programación X-Windows: el acelerador gráfico

8 413042 303299

SÓLO PROGRAMADORES

Número 39
SÓLO PROGRAMADORES
es una publicación de
TOWER COMMUNICATIONS

Director Editor
Antonio M. Ferrer Abelló
aferrer@towercom.es
Directora Adjunta
Amelia Noguera
anoguera@towercom.es
Colaboradores
E. Díaz, Manuel de la Herrán, E. de la Lastra, Eugenio Castillo, Oscar Prieto Blanco, Ernesto Schmitz, Alejandro Reyero, Chema Álvarez, J. Antonio Mendoza, Jorge Delgado Mendoza, M. Huertas.
Jefe de Diseño
Fernando García Santamaría
Maquetación
Clara Francés
Tratamiento de Imagen
María Arce Giménez
Imagen de Portada
Fernando Núñez

Publicidad
Erika de la Riva (Madrid)
Tel.: (91) 661 42 11
Pepín Gallardo (Barcelona)
Tel.: (93) 213 42 29

Suscripciones
Isabel Bojo
Tel. (91) 661 42 11 Fax: (91) 661 43 86
suscrip@towercom.es

Laboratorio
Óscar Rodríguez (jefe)
Javier Amado
Servicio Técnico
Óscar Casado

Filmación
Megatipo
Impresión
G. Reunidas
Distribución
SGEL

TOWER COMMUNICATIONS
Director General
Antonio M. Ferrer Abelló
Director Financiero
Francisco García Díaz de Liaño
Director de Producción
Carlos Peropadre
Directora Comercial
Carmina Ferrer
carmina@towercom.es

Redacción, Publicidad y Administración
C/ Aragoneses, 7
28108 Pol. Ind. Alcobendas (MADRID)
Telf.: (91) 661 42 11 / Fax: (91) 661 43 86

Depósito legal: M-26827-1994
ISSN: 1134-4792
PRINTED IN SPAIN

EDITORIAL

Poderoso caballero es don dinero

No es un negocio sucio, pero no debemos olvidar que es un negocio muy lucrativo. La informática no transcurre por los bucólicos senderos de la Verdad y el Avance auténticos, ésos que se deben sólo a sí mismos y no a razones particulares y temporales, sino por los tortuosos caminos de la verdad y el avance interesados, ésos que se deben a don Dinero, el poderoso caballero que ha sabido sobrevivir desde el Siglo de Oro. Por eso, que dos de las empresas más importantes del sector se enzarcen en una batalla zafia en la que salen a relucir operaciones o, por lo menos, intereses poco claros, no nos sorprende, pero sí nos produce un poco de tristeza. Por un momento, en el calor de la lucha, se han quitado las máscaras del progreso para mostrar sus íntimas entrañas, doradas pero frías y calculadoras en exceso. Si la razón es de Sun o de Microsoft es algo en lo que de momento no vamos a entrar, pero es triste observar a dos colosos tirándose de los pelos como chiquillos mientras los demás esperamos para actuar en consecuencia.

Dejemos de lado los conflictos del sector para centrarnos en ese futuro cercano que todavía nos pertenece. Como sabéis, del 4 al 9 de Noviembre se celebrará en Madrid, como todos los años, la tradicional feria SIMO TCI, a la que acuden las más importantes empresas del sector y numerosos profesionales que quieren estar al día en nuevas tecnologías y productos. Con este motivo lanzaremos en esas fechas un número especial íntegramente dedicado a tecnologías de programación en Internet que no os podéis perder. ActiveX, CGI y ASP, Arquitecturas Cliente-Servidor, VisualAge for Java, y mucho, mucho más será lo que podáis encontrar en este número extra sobre Internet.

Esperando veros en el SIMO os dejamos saboreando el ejemplar que tenéis en vuestras manos, en el que se incluyen temas para todos los gustos, desde la teoría pura y dura del diseño y análisis OO, pasando por la aventura de crear nuestra propia Intranet con vuestro SO favorito, el Linux, hasta llegar a la isla del relajante ajedrez, donde aprenderemos un poco más sobre IA aplicada a un dominio de conocimiento concreto. Y no dejéis de lado el CD-ROM, cada día con mejores contenidos. Seguro que encontráis algo que os interesa entre sus muchísimos megas de productos.

Tenéis a vuestra disposición la dirección de correo electrónico de la revista: solop@towercom.es, donde podéis enviar vuestras sugerencias sobre los contenidos, los temas que queráis que incluyamos y los productos que os gustaría que distribuyéramos con el CD-ROM, que todos los meses regalamos. Sois nuestros mejores colaboradores.

SÓLO PROGRAMADORES



Noticias

Deep Blue Jr. hace estragos VENCE A LA MAYORÍA DE SUS CONTRINCANTES

Fueron muchos los que vieron cómo Deep Blue Jr. utilizaba artimañas de jugador de ajedrez para vencer en casi todas las partidas que se jugaron el pasado 7 de octubre en la sede de IBM en España. Tan sólo no pudo con uno de los mejores ajedrecistas españoles PAco Vallejo, quien, con tan sólo quince años, resultó vencedor.

Deep Blue Jr. es una versión adaptada de la máquina que en mayo venció al campeón del mundo de ajedrez, Gary Kasparov. Esta versión 'más inmadura' dispone de dieciséis procesadores y puede calcular diez millones de posiciones de ajedrez por segundo, mientras que una persona, con muchas tablas, puede llegar a las dos jugadas por segundo.

Las dos versiones de Deep Blue utilizan el ordenador de IBM RS/6000, que ya tiene mucha experiencia en esto de ser utilizado en proyectos de este calibre, no en vano fue el elegido para participar en el sistema informático que llevaba la nave Pathfinder que indagó en los secretos de Marte.

Las vísceras de Deep Blue tienen mucho de sus creadores, el equipo de investigación del Centro de Investigación Thomas J. Watson de IBM en Nueva York, quienes utilizaron unos de los lenguajes con más adeptos dentro de la informática más pura, el lenguaje C, construyendo un sistema paralelo y muy escalable capaz de jugar en tres minutos (el tiempo medio asignado a cada jugador para realizar un movimiento).

Además Deep Blue tiene una base de datos con las mejores partidas de los últimos cien años y otra base de datos con los finales de partida.

Este proyecto de investigación que puede resultar para algunos un esfuerzo vano en el que se invierten muchos millones de pesetas, tiene sin embargo campos de aplicación de extrema importancia, como la simulación de dinámica molecular que se utiliza en la industria farmacéutica, donde el tiempo que se emplea para lanzar un nuevo fármaco al mercado es de unos doce años y conlleva unos costes muy elevados, si se aplica la tecnología utilizada en el Deep Blue, el coste y el tiempo de lanzamiento se reducirán a la mitad (en algunos casos) lo cual supone una importante mejora. Además esta tecnología se puede aplicar también en los sistemas de análisis de mercados financieros o el control de tráfico aéreo de los principales y más congestionados aeropuertos internacionales.

El objetivo que persigue IBM es aprovechar la experiencia sin precedentes que se está obteniendo con el desarrollo de este proyecto para resolver problemas de extrema complejidad que puedan presentarse en un futuro.

Esperamos que tanto esfuerzo en este sentido obtenga sus frutos pronto, y este tipo de noticias nos alegren más que nos sorprendan.

La taza de café siembra la discordia

SUN DEMANDA A MICROSOFT

Muchos lo veían venir desde hace tiempo, pero no creemos que pensaran que la cosa iva a terminar adquiriendo estos tintes de película estadounidense. Sun Mycrosistems, Inc. ha interpuesto una demanda contra Microsoft Corporation en la Corte de Distrito de EEUU por los siguientes argumentos: violación de marca registrada, publicidad falsa, incumplimiento contractual, competencia desleal, interferencia con posibilidad de ventajas económicas e inducción al incumplimiento de contrato.

El tema no tiene desperdicio: Microsoft presentó a primeros de octubre la disposición del navegador Internet Explorer 4.0 y, una vez evaluado, desde Sun se considera que este producto, además de otros que como éste llevan el logotipo Java Compatible como puede ser el Kit de Desarrollo de Software Java (SDKJ), no han superado las pruebas de compatibilidad Java, que determinan si las tecnologías utilizadas se acogen a las especificaciones e interfaces de programación Java.

Ésta es un de las obligaciones contractuales que cualquier licenciatario de Java debe cumplir.

Esta falta de compatibilidad origina que las aplicaciones que se desarrollen utilizando las herramientas de Microsoft no compatibles no puedan ejecutarse (o lo hagan con problemas) sobre otros sistemas operativos como MacOs o Unix, o sobre navegadores como Netscape Navigator, y a la inversa, las aplicaciones desarrolladas sobre el JDK no podrán funcionar sobre Internet Explorer 4.0.

La demanda se refiere al aspecto técnico en los siguientes términos: "... En particular, el acusado Microsoft ha modificado deliberadamente las interfaces de programación de Java (APIs)... mediante la adición secreta de especificaciones Win32 y otras APIs, a las bibliotecas de clases Java... Microsoft ha actuado para inducir a los desarrolladores independientes de software que utilizan el SDJK a escribir programas de forma inadvertida... esperando lograr funcionalidad multi-plataforma...".

Desde Microsoft se indica que la demanda de Sun es injuriosa y afirman que Microsoft ha proporcionado la implantación más compatible de Java del Mercado. IE 4.0 proporciona la integración de Java más funcional y compatible del mercado. Según nota de prensa llegada a esta redacción: "Sun aparentemente ha decidido que no puede competir en el mercado. Pensamos que las evidencias demostrarán que mientras que Microsoft ha cumplido íntegramente su acuerdo con Sun, Sun ha incumplido reiteradamente sus obligaciones con Microsoft. Debido a que este tema está en los tribunales, Microsoft no puede dar más detalles sobre el caso".

Mientras tanto, los programadores de todo el mundo esperan ansiosos: ¿aprendo Java o me olvido de Java?, y los analistas dudan: ¿utilizo en este proyecto Java o mejor me olvido y sigo con C++?.

Desde la redacción de Sólo Programadores podemos decir que nos gustaría que este tema no tuviera visos de llegar a convertirse en una lucha sin ningún tipo de razón de ser ni ganadores, todos perdemos cuando lo que está en el frente de batalla es la libre circulación de nuevos elementos dentro del mercado de la informática. Existen muchas personas, quizá demasiadas, que invierten mucho de su tiempo en el estudio de las novedades que van apareciendo. Citamos las palabras de un consultor que decía que en su consultoría no daban a basto a hacer horas extras para estudiar los productos nuevos, porque las nuevas versiones salían antes de que conocieran las anteriores. Si añadimos este tipo de intereses al asunto, estamos apañados.

Breves

WINDOWS CE ENTRE NOSOTROS
Microsoft presenta el SO para ordenadores empotrados

Este Sistema operativo se concibe para ser una plataforma que potencie el desarrollo de una nueva gama de aplicaciones informáticas, como los Handheld PCs, las consolas de juegos, teléfonos inteligentes, reproductoes de DVD, y hasta electrodomésticos, además de las habituales como el control de procesos, instrumentación, etc.

Windows CE proporciona soporte para la ejecución de tareas en tiempo real, la ejecución de programas de mayor tamaño que la memoria RAM del sistema, soporte completo para UNICODE, soporte de fuentes TrueType para definir interfaces de usuario final de alta calidad, etc.

Las arquitecturas y procesadores de 32 bits soportados son ARM, MIPS, PowerPC, StronARM, SuperH y x86 de AMD, Digital, Hitachi, Intel, Motorola, Nec, Philips y Toshiba.

PROBLEMAS CON EL EURO: ¿quién será capaz de llegar al 2000?

Para los profesionales de las tecnologías de la información la llegada de la Unión Monetaria Europea representa un reto que tendrá que ser superado en los pocos años que nos quedan hasta que tal hecho tenga lugar. Numerosas empresas nos proponen ya soluciones que se enticipen al evento. PeopleSoft es una de ellas. Su arquitectura funcional global multimoneda permite establecer múltiples contabilidades y planes contables que constituirán la base para la preparación y transición hacia la EMU. Para más información: http://www.peoplesoft.com.

MAYOR OFERTA EN SISTEMAS UNIX

IBM anuncia la comercialización de sistemas UNIX con tecnología de 64 bits

Se espera mucho de IBM e IBM no suele defraudar; desde la compañía se conoce este sentimiento en el mercado y se trabaja duro para estar a la altura.

O por lo menos eso parece demostrar la nueva gama de productos que recientemente presentó, entre los que destaca la nueva solución para sistemas UNIX con tecnología de 64 bits, que incluye un servidor IBM RS/6000 modelo S70, la versión 4.3 de AIX, que permite a los usuarios utilizar aplicaciones de 64 y 32 bits y hace gala de ser la más rápida para operaciones Web de todo el entorno UNIX, y un conjunto de aplicaciones de 64-bits especialmente desarrolladas para aprovechar la máxima funcionalidad de este equipo.

La versión de 64 bits está pensada para que los sistemas puedan incrementar la velocidad de proceso, y fomentarán la utilidad de las nuevas aplicaciones, lo que engloba aplicaciones de datamining y datawarehousing que tienen requerimientos muy concretos en cuanto a los sistemas operativos, que deben ser capaces de manejar bases de datos mucho más grandes de lo habitual.

Además, IBM anuncia los modelos 397 y 43P de RS/6000, los adaptadores de entorno gráfico GXT 800M y GXT 120P, el Thin Node P2SC y el nuevo entorno operativo RS/6000 en CD-ROM. El conjunto de estaciones de trabajo y servidores RS/6000 está diseñado para mejorar la productividad dentro del entorno científico-técnico, en el ámbito de la investigación científica y el diseño industrial.

Así, estas herramientas tienen muy buena aceptación en universidades y departamentos de investigación de empresas que invierten mucho esfuerzo y dinero en I+D. Es de resaltar que el mayor telescopio de Estados Unidos diseñado por la Universidad de Texas fue construido utilizando este tipo de estaciones de IBM, en particular la RS/6000.

Así, no es de extrañar que Deep Blue pueda llegar a vencer a Kasparov, y a otros como él.

NACE AURORA

El futuro componente de Nestcape Communicator

Bajo este nombre clave, la compañía Netscape ofrecerá a los usuarios un nuevo componente que se va a servir de la tecnología de software cliente de próxima generación, la cual permitirá integrar contenidos de Internet y del escritorio del usuario.

Aurora permitirá a todos los usuarios disponer de una interfaz que sea de fácil manejo, así como capaz de integrar y gestionar la información que provenga de las distintas fuentes. Las direcciones Web y los canales de distribución, las bases de datos, el correo electrónico e incluso los documentos y archivos que se encuentren situados en el interior del escritorio del usuario podrán ser accesibles fácilmente sin necesidad de actualizar el SO.

Aurora será un programa totalmente compatible con las principales plataformas de aplicaciones existentes y permitirá utilizar el RDF (*Resource Description Framework*), quizá el futuro estándar para realizar la gestión de metadatos (se trata de estructuras de datos específicas dentro de las que es posible contener toda la información necesaria sobre la organización en contenido en aplicaciones Internet, intranet, extranet, y escritorio).

RDF se ha basado en XML (*eXtensible Mark-up Language*), un lenguaje de próxima generación que ha sido específicamente diseñado para el diseño en la Web.

Las aplicaciones en las que será posible utilizar este nuevo componente van desde las que utilizan mapas de direcciones para facilitar la navegación, las de indicación de direcciones de Internet o de intranets, los sistemas de clasificación y restricción de contenidos, las definiciones de canales de distribución automática, etc., hasta las firmas digitales, la recogida de bibliotecas digitales, etc.

RDF se puede utilizar para describir perfectamente otros conjuntos de información. En esta ocasión es utilizada por Aurora para ofrecer una interfaz multiplataforma con la que sea posible ver, navegar y gestionar la información y unificar dicha información sin tener en cuenta los lugares de los que proceda.

RDF es, en definitiva, la materialización de la propuesta que la compañía Netscape realizó para definir una nueva infraestructura que permitiera tanto la manipulación como la asociación de información que esté interconectada a través de una red determinada.

COMPUTER ASSOCIATES ofrece gestión Java madiante la interfaz de navegador Unicenter TNG

Ajena a los problemas que pueden derivarse del enfrentamiento entre Sun y Microsystemas, importantes empresas del sector continúan apostando por este lenguaje y su entorno de desarrollo. Unicenter TNG es una solución para la gestión empresarial integrada que permite a las empresas gestionar todo tipo de recursos de tecnologías de la información. El producto suministra gestión empresarial global para redes TCP/IP, SNA, IPX/SPX y DECnet y 40 plataformas, como PCs, Unix, AS/Net Ware, Windows NW y entornos mainframe.

El producto realiza diferentes tareas como son la búsqueda de redes, topología, rendimiento, eventos y estado, seguridad, distribución de software, carga de trabajo, soporte para el cambio de gestión y otras funciones para entornos informáticos distribuidos, internet e intranets. Además, la tecnología agente de Unicenter TNG gestiona los sistemas de Java.

Para más información sobre este u otros productos de CA se puede consultar su dirección de correo electrónico http://www.ca.com.

Java y UML componen el núcleo de las herramientas de desarrollo de ORACLE

Oracle ha presentado recientemente su nueva estrategia que amplía su gama de herramientas para desarrollar aplicaciones para networking. Y Java vuelve a ser protagonista. No sabemos qué grado de utilización en la industria informática terminará teniendo este lenguaje, a tenor de los acontecimientos, pero desde luego ocupa un lugar privilegiado en la mayoría de los departamentos de desarrollo de las principales empresas del sector.

En este sentido, Oracle ha creado un conjunto de herramientas para la actualización, adaptación y desarrollo de objetos comerciales Java, soportando Java en la parte cliente, en el servidor de aplicaciones y en el servidor de datos. Dentro de estas herramientas se incluye nuevas versiones de Designer/2000, Developer/2000 y una nueva herramienta de desarrollo Java. Con ellas, se podrá trasladar códigos entre plataformas, conseguir un equilibrio de cargas y tolerancia a fallos, reutilizar componentes y aprovechar los medios e instalaciones de la plataforma de proceso en red, utilizando un lenguaje estándar que es accesible a todos los desarrolladores. Y la forma de hacer todo esto es sencillamente mediante la utilización de los objetos Java, ya que separan el modelo de objetos de la forma en que se presenta la interfaz de usuario. El modelo de objetos incluye una abstracción para cada objeto, reglas comerciales, relaciones entre objetos, múltiples interfaces de usuario para cada objeto e información sobre almacenamiento de objetos. Estos nuevos modelos se crean para ser utilizados en las aplicaciones y se representan en *Enterprise JavaBeans*, por lo que pueden ser utilizados por todo tipo de herramientas.

Esta nueva tecnología recibe el nombre de *Objetos Comerciales Java* o *Java Business Objects*. El entorno de desarrollo proporciona herramientas de modelización de objetos basadas en el UML (*Unified Modeling Lenguage*), significando de esta forma un importante avance en lo que se refiere a herramientas de modelización.

Breves

OPTIMIZACIÓN DE SOLARIS. Sun orienta el SO hacia la Web

Con Solaris Millenium, se pretende dirigirse al mercado de los profesionales de la tecnología de la información, directores de líneas de negocios, proveedores de servicios Internet (ISP) y usuarios de equipos de sobremesa de gran potencia.

La versión con mejorar para Web se ha creado para hacer frente a las nuevas necesidades de los clientes, que incluyen trabajo en intranet, Internet, e informática empresarial. Entre sus nuevas funcionalidades destacamos el High Availability Clustering que implementa la nueva tecnología de clusters de Sun, Full Moon, para informática crítica y de bases de datos.

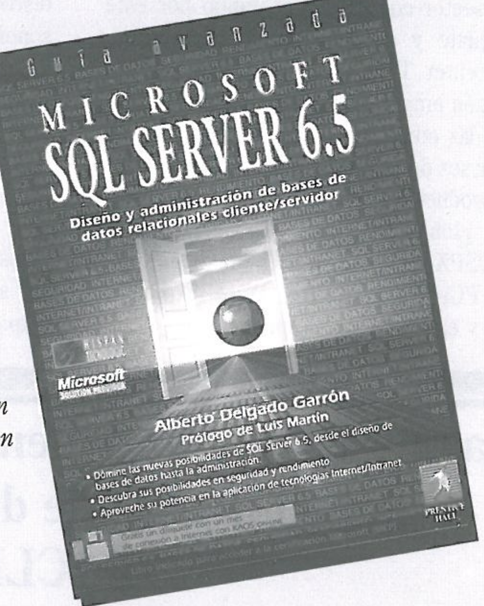
MICROSOFT PRESENTA EXPLORER 4.0 CON IMPORTANTES INNOVACIONES

Que Microsoft Explorer 4.0 es un producto esperado por muchos y por distintas razones, todos lo sabemos. Ahora podemos comprobar si las expectativas son fundamentadas o no, y rezar para que nuestra máquina (PC) pueda con ello.

Las innovaciones que incorpora IE 4.0, como las tecnologías de Canal Activo (Active Channel) y el nuevo Escritorio Activo, por citar algunas, permiten a los fabricantes de PCs ofrecer a sus clientes nuevas funcionalidades y servicios, como mejor soporte técnico, suministro actualizado de información especializada, actualizaciones de producto, etc. Pero para conocerlo en profundidad, nada mejor que instalarlo en nuestro equipo, en el CD-ROM podéis encontrarlo.

Microsoft SQL Server 6.5, Guía Avanzada

Este libro está especialmente indicado para aquellos profesionales de la informática que deseen acceder a las certificaciones de Microsoft MCP (*Microsoft Certified Professional*). Se divide en dos apartados: Diseño y codificación de bases de datos relacionales y Administración de SQL Server 6.5. La primera introduce al lector en los conceptos relacionados con la Teoría de Bases de Datos Relacionales y presenta los elementos principales del lenguaje SQL (consultas, sentencias de modificación y definición de datos, etc.). La segunda parte trata sobre la administración de datos en SQL Server 6.5, tratando en profundidad temas como la manipulación y administración de la seguridad en el acceso a los datos o las técnicas para la creación de objetos de bases de datos y sus estructuras de almacenamiento.

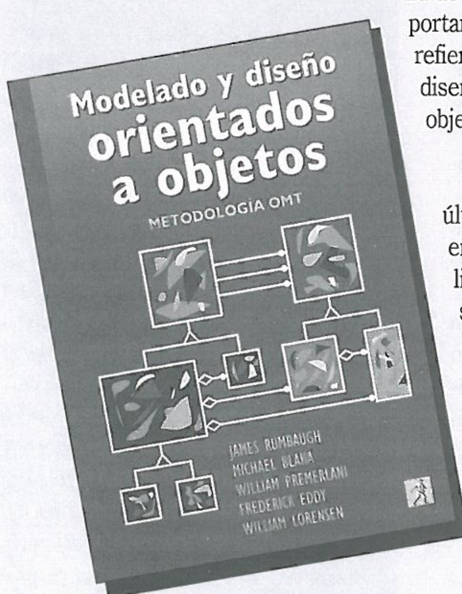


Su autor, Alberto Delgado Garrón, es director de estudios de *Hispan Technology*, centro de formación informática avanzada y *Microsoft Certified Solution Provider* y está avalada por Microsoft. Por tanto, un libro de excelente calidad.

Editorial: Prentice Hall, 1997
Nº de páginas: 496
PVP: 4.000 pts.
Autor: Alberto Delgado Garrón
Idioma: Español
Nivel: Avanzado

Modelado y diseño orientado a objetos

La de los noventa ha sido una década en la que el desarrollo de software ha dado un giro importante en cuanto a metodología y tecnología. Si pudiéramos resumir el ahora en lo que se refiere a la informática tendríamos que hablar de internet, si hablamos de metodologías de diseño y programación deberíamos hacer un estudio sobre el paradigma de la orientación a objetos. Todo es OO últimamente pero, ¿quiénes son los padres de la criatura?



Uno de los libros que tratan sobre el diseño OO que ha sufrido más consultas en los últimos años ha sido, sin ninguna duda, el que en este número comentamos. Próxima se encuentra, según algunos, la aparición de nuevos estándares que mejorarán los ahora utilizados. Pero antes de que esto ocurra, aquellas personas que desde las empresas de desarrollo se dedican a trabajar utilizando los enfoques y también las herramientas OO, deberían conocer con precisión cuáles son las bases, y OMT es en este momento una de las metodologías y notaciones gráficas más extendidas entre los analistas y programadores. Sólo indicar que si la OO nos parece interesante, cuando menos es preciso mantenerlo entre aquellos libros que tienen que estar localizados en nuestra biblioteca favorita o nuestra estantería.

Editorial: Prentice Hall
Nº de páginas: 643
Nivel: Avanzado
Autores: Rumbaugh y otros
Idioma: Español
Precio: 4.000

Netscape Netcaster y JavaScript 1.2

Alejandro M. Reyero Abad
axl@las.es

En este artículo aprenderemos a utilizar nuevo código JavaScript y HTML, necesario para el mantenimiento de un canal Netcaster, ya sea normal o *webtop*.

En el artículo del número pasado vimos cómo se crean los canales, los diferentes tipos de canal y los pasos necesarios para que un usuario se suscriba a nuestro canal desde una página WEB. Ahora es tiempo para el desarrollo del canal, por lo que veremos algunos conceptos, hasta hoy nuevos, pero que es necesario que conozcamos para sacar todo el partido de Netscape, como las paletas flotantes, el refresco automático, la ayuda sensible al contexto, etc.

Paletas Flotantes

Las paletas flotantes (desde donde el usuario controla el comportamiento del canal webtop) permanecen en primer plano en nuestro escritorio, precediendo al resto de las ventanas. De esta manera, la funcionalidad del webtop está siempre disponible para el usuario y nunca es ocultada por otras aplicaciones (los programadores windows las llaman ventanas modales).

En la tabla llamada "flotante.htm" podemos encontrar el código necesario para que una página (en este caso el WEB Todo Juegos) sea la ventana principal en la jerarquía Z-Order, asignando el valor *"yes"* a la propiedad *"allwaysRaised"* al abrir la nueva ventana, lo que hará que la ventana permanezca siempre en primer plano.

Flotante.htm

```
<HTML>
<HEAD>
<TITLE>
Ejemplo de Jerarquia Z-Order
</TITLE>
<SCRIPT LANGUAGE = "JavaScript1.2">
var objetivo=self;
var nuevaVentana=null;
var pixel,color;
var width = screen.width;
var height = screen.height;
function abrirVentana() {
netscape.security.PrivilegeManager.enablePrivilege
                              ("CanvasAccess");
nuevaVentana=
window.open("http://todojuegos.las.es","test",
"left=210,top=10,outerWidth=500,
outerHeight=500,location=no,titlebar=yes,
toolbar=no,alwaysRaised=yes");
objetivo=nuevaVentana;
netscape.security.PrivilegeManager.revertPrivilege(
"CanvasAccess");
}
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE=BUTTON
onClick="abrirVentana()"
VALUE="Abrir Nueva Ventana">
</FORM>
</BODY>
</HTML>
```

La creación de canales PUSH para Netcaster supone la utilización, para su control y presentación visual, de potentes y nuevas (o remozadas) instrucciones JavaScript.

Algunas de las funciones JavaScript utilizadas en el ejemplo *flotante.htm* son protegidas, requiriendo para su utilización código con firma (*signed code*), el cual veremos con detenimiento más adelante.

Pantalla Completa

El usuario percibe el canal webtop como un escritorio de trabajo debido a que éste ocupa la pantalla completamente. Esto se consigue con el modo "canvas" de Netscape Communicator.

La mayoría de los desarrolladores de canales webtop no tendrán que preocuparse por la implementación directa de la característica canvas de Communicator, ya que Netcaster abre el canal en el modo especificado en las propiedades del objeto canal (como vimos en el artículo del mes pasado).

Para utilizar el modo canvas, abriremos la ventana con el código JavaScript que aparece en flotante.htm, tres simples líneas que abren una ventana a pantalla completa, sin controles, bordes, menús ni barras de desplazamiento.

Tabla 1

```
netscape.security.PrivilegeManager.enable
Privilege("CanvasAccess");
window.open("http://home.netscape.com","
home",'titlebar=no,screenX=-1, screenY=- 1,
toolbar=no,scrollbars=no,menubar=no
outerWidth=screen.availWidth,outerHeight
=screen.availHeight,alwaysLowered=yes');
netscape.security.PrivilegeManager.revert
Privilege("CanvasAccess");
```

Arrastrar y Soltar

Las nuevas capacidades de JavaScript 1.2, como son las capas (layers) y hojas de estilo (las cuales vimos con detalle en los artículos publicados en los números 36 y 37 de Sólo Programadores), combinadas con el modelo de eventos JavaScript, nos permiten desarrollar canales webtop con un control absoluto sobre los contenidos y continentes. Las imágenes podrán ser colocadas en cualquier lugar de la pantalla y sus posiciones grabadas.

Las nuevas capacidades de JavaScript 1.2 nos permiten colocar imágenes en cualquier lugar de la pantalla

El siguiente código HTML define una capa con una imagen como su contenido. Esta capa incluye un fichero JavaScript que implementa capacidades Arrastrar y Soltar (Drag & Drop) a la misma.

```
<LAYER NAME="nuestra_capa"
LEFT=300 TOP=150>
<IMG SRC="foto.jpg">
<SCRIPT LANGUAGE="JavaScript1.2">
SRC="codigo.js">
</SCRIPT>
</LAYER>
```

El fichero JavaScript que proporcionará la capacidad de Arrastrar y Soltar a nuestra capa, *codigo.js*, es el siguiente:

```
var viejaX, viejaY;
function empiezaArrastra(e) {
  document.captureEvents(Event.MOUSEMOVE);
  document.onMouseMove=drag;
  viejaX=e.pageX;
  viejaY=e.pageY;
  return false;
}
function finArrastra(e) {
  document.onMouseMove=0;
  document.releaseEvents(Event.MOUSEMOVE);
  return false;
}
function drag(e) {
  moveBy(e.pageX - viejaX, e.pageY - viejaY);
  viejaX = e.pageX;
  viejaY = e.pageY;
}
document.captureEvents(Event.MOUSEUP|
                       Event.MOUSEDOWN);
document.onMouseDown=empiezaArrastra;
document.onMouseUp=finArrastra;
```

Más adelante veremos con detenimiento los nuevos eventos JavaScript.

Refresco Automático

Una de las ventajas de los canales sobre las páginas WEB es la capacidad de enviarle al usuario contenidos personalizados dinámicamente. Parcialmente, se logra mediante los "parámetros de netcasting".

Estos parámetros son valores de configuración que controlan las acciones requeridas por los canales para mandar la información al usuario a unas determinadas horas, así como la información que le ha de enviar. Los valores por defecto de estos parámetros se dan a través de las propiedades del objeto canal, el cual vimos en el artículo anterior.

La arquitectura NetHelp de Communicator nos permite añadir ayuda sensible al contexto a nuestros canales

Ayuda Sensible al Contexto

La nueva arquitectura NetHelp, que incluye Communicator, permite que nuestros canales tengan una ayuda sensible

al contexto, permitiendo informar al usuario en todo momento sobre las acciones necesarias para navegar por nuestro canal.

Para conocer con más detalle la ayuda NetHelp, podemos visitar la página de Netscape http://home.netscape.com/eng/help.

JavaScript 1.2 y Netcaster

Veremos a continuación las extensiones al lenguaje JavaScript que nos permiten utilizar algunas de las capacidades de los canales, incluyendo el poder ejecutarse en modo webtop, compatibilidad con versiones anteriores de JavaScript, operadores, propiedades, métodos, objetos y eventos.

Como vimos en el primer ejemplo de este artículo, algunas de las nuevas opciones necesitan utilizar JavaScript firmado debido a razones de seguridad (lo veremos más adelante). Cada una de las opciones que requieren el uso de JavaScript firmado para su utilización aparecerá como "Protegida" a partir de ahora.

Compatibilidad con Versiones Anteriores

La evolución de JavaScript supone grandes ventajas para los desarrolladores y usuarios, pero puede provocar que código escrito para Netscape Communicator 4.0 no funcione, o se comporte de una manera extraña, en versiones anteriores.

Para evitar los problemas con los usuarios de versiones anteriores, es muy conveniente utilizar el atributo *LANGUAJE* en el TAG *<SCRIPT>* para indicar la versión de JavaScript que estamos utilizando.

El atributo Languaje indica qué versión de JavaScript debe soportar el browser

Todo el código incluido entre TAGs *<SCRIPT>* es ignorado por el browser que no tiene el nivel de soporte JavaScript indicado en el atributo *LANGUAJE*. Por ejemplo, todo el código incluido dentro del TAG *<SCRIPT LANGUAJE="JavaScript1.1>* será ignorado por Navigator 2.0 y utilizado por Navigator 3.0 y 4.0.

Operadores

Algunos de los operadores vistos en artículos anteriores han cambiado su comportamiento o son nuevos.

Operadores de Igualdad

Si utilizamos el TAG <SCRIPT LANGUAGE="JavaScript1.2">, Navigator (y Netcaster) interpretará los operadores de igualdad (== y !=) de distinta manera que en anteriores versiones. Los nuevos operadores de igualdad se comportan de la siguiente manera:

- Nunca intentan convertir operandos de un tipo a otro. Tendremos que convertirlos explícitamente.
- Siempre comparan operandos de idéntico tipo. Operandos de distinto tipo son no iguales.

Estos cambios se introdujeron para simplificar el lenguaje, haciendo que JavaScript se parezca a otros lenguajes en la manera de interpretar los operadores de igualdad, lo que ayuda a evitar errores.

Operador delete

JavaScript 1.2 introduce un nuevo operador, *delete*, que se encarga de borrar un objeto, una propiedad de un objeto o un elemento en un array:

```
delete nombreObjeto
delete nombreObjeto.nombrePropiedad
delete nombreArray[indice]
```

Si la operación *delete* es culminada con éxito se asigna al objeto, propiedad, o elemento el valor "indefinido", devolviendo el booleano *true*; si falla, devuelve *false*.

Propiedades

Al igual que para los operadores, JavaScript 1.2 tiene nuevas propiedades, así como algunos cuyo comportamiento ha sido modificado.

JavaScript 1.2 tiene nuevas propiedades, eventos, operadores y métodos, así como modificaciones a los ya conocidos

Propiedades navigator

Vemos a continuación las nuevas propiedades del objeto *navigator*:

- **languaje:** Indica la nacionalidad de la versión de Navigator (o Netcaster) que estamos utilizando. Su sintaxis es: *navigator.languaje*. El valor que retorna es un código compuesto de dos letras que indica

el lenguaje del Navigator que está en ejecución.

- **platform:** Indica el tipo de máquina en el que la versión de Navigator que se está ejecutando fue compilada. Su sintaxis es:*navigator.platform.*

- **Propiedades window**

- **innerHeight:** Especifica la dimensión vertical, en pixeles, de la ventana contenido. Su sintaxis es *[ventana.]innerHeight.* Si queremos crear una ventana menor de 100 por 100 pixeles tendremos que utilizarla en un script con firma.
- **innerWidth:** Exactamente igual que el anterior pero especificando la dimensión horizontal.
- **locationbar:** Indica si la barra de navegación de la ventana Netscape está visible o no. Su sintaxis es

  [ventana.]locationbar.visible= true|false.

  Para utilizar esta propiedad tendremos que utilizar scripts firmados.
- **menubar:** Igual que el anterior pero para la barra del menú.
- **personalbar:** Igual que las anteriores pero para la barra de herramientas "personal"
- **scrollbars:** Como las anteriores pero para las barras de desplazamiento.
- **statusbar:** Una más, en esta ocasión para la barra de estatus.

Métodos

Veremos a continuación algunos de los nuevos métodos incluidos en JavaScript 1.2.

- **back:** Hace que Navigator vuelva a la dirección internet (URL) anteriormente visitada (equivalente a la pulsación del botón *Back*). Su sintaxis es:: *[ventana.]back.*
- **forward:** Igual que el anterior pero va a la siguiente dirección (equivalente a la pulsación del botón *Forward*)
- **print:** Imprime los contenidos de la ventana especificada. Equivalente a pulsar el botón *Print* en la ventana Navigator.
- **stop:** Detiene el *download* en progreso en la ventana indicada. Equivale al botón *Stop.*
- **captureEvents:** Permite que la ventana o documento capture eventos del tipo especificado. Su sintaxis es:

  [objeto.]captureEvents(Event.tipo Evento).

- **find:** Busca el texto especificado entre los contenidos de la ventana actual. Su sintaxis sería:

  [ventana.]find(["texto"][,true|false][,true|false]).

  Devuelve *true* si lo encuentra o *false* en caso contrario. Los dos parámetros booleanos son optativos, el primero hace que la búsqueda distinga entre mayúsculas y minúsculas si es *true* y el segundo provoca que la búsqueda se realice de atrás hacia delante.
- **toString:** Convierte un objeto o un array en una cadena. Su sintaxis es: *objeto.toString().*

Eventos

En este momento ya podíamos imaginar que no podía ser de otra manera, JavaScript incluye nuevos eventos junto con los ya conocidos, algunos de los cuales han sido a su vez modificados.

- **Click:** Indica que se han producido los eventos *mouseDown* y *mouseUp* sobre un link o botón. Su sintaxis es similar a la de los métodos mencionados, esto es: *onClick="codigo JavaScript".*
- **DblClick:** Igual que el anterior pero para una doble pulsación.
- **DragDrop:** Indica que un objeto ha sido soltado sobre la ventana en cuestión.
  Su sintaxis es:

  onDragDrop="codigo JavaScript".

Con el tag <script languaje="JavaScript1.2">, Navigator interpretará los operadores de igualdad de distinta forma que en anteriores versiones

- **KeyDown:** Igual que los anteriores pero indica la pulsación de una tecla.
- **KeyPress:** Indica que una tecla está pulsada.
- **KeyUp:** Indica que la tecla ha sido soltada.
- **MouseMove:** Indica que el ratón ha sido movido.
- **MouseOut:** Indica que el ratón ha sido movido fuera del área del objeto al que hacemos referencia.
- **MouseOver:** Indica que el ratón está sobre el objeto en cuestión.
- **Resize:** Indica que la ventana ha cambiado de tamaño.

Modelo Netscape de Capacidades

Como ya hemos comentado, es posible que algunos de los programas Java Y JavaScript que nos bajamos con un canal necesiten estar firmados digitalmente para ser utilizados. Esta característica forma parte del modelo de seguridad de JavaScript, conocido como "Modelo Netscape de Capacidades". La distribución de software a través de redes como internet supone un riesgo para el usua-

rio, ya que esos programas pueden intentar modificar el comportamiento de nuestro ordenador, borrar datos que pudieran ser útiles, obtener información confidencial, etc.

Hasta JavaScript 1.2, la manera que tenía Netscape de asegurar que no se iban a producir daños irreparables para el usuario era no dejar que se ejecutaran programas en el ordenador del usuario. No obstante, algunas de las posibilidades de JavaScript y los applets Java se veían mermadas, así que JavaScript 1.2 establece un nuevo modelo de seguridad permitiendo que determinadas acciones, antes prohibidas, se ejecuten en determinadas circunstancias o con el consentimiento del usuario.

JavaScript 1.2 establece un nuevo modelo de seguridad que pide autorización al usuario antes de realizar acciones antes prohibidas

El modelo de capacidades de Netscape está basado en un protocolo de firma de objetos que permite al webmaster crear aplicaciones que piden permiso al usuario para acceder a los recursos de su sistema.

Este protocolo permite al usuario escoger el tipo de acceso que va a permitir a su sistema (no "todo o nada"), limita ese acceso a las líneas de código que necesiten protección y también limita la duración del acceso a la duración de la llamada que lo necesita.

Este proceso permite al usuario decidir qué software se va a bajar a su disco duro, de la misma manera que escoge los programas que se ejecutarán en él. Una vez que el usuario ha decidido que el proveedor de información es, en cierto modo, "fiable", se instala en su copia de Communicator un certificado criptográfico para cada proveedor.

Después, Communicator ya podrá utilizar esa misma firma digital que ha sido previamente comprobada para determinar si la información que le está llegando es de uno de los proveedores autorizados.

Cuando un objeto firmado ha sido bajado, Communicator presenta al usuario una caja de dialogo que indica quién ha firmado el objeto y qué necesita ejecutar en su ordenador. Entonces el usuario puede autorizar al proveedor para que Communicator no vuelva a pedir en ningún otro momento después autorización para la ejecución del código del proveedor.

Certificados y Firmas Digitales

Un certificado es un documento electrónico que es utilizado para identificar a alguien.

Un certificado con firma es una clase especial de certificado que nos permite asociar nuestra firma digital con cualquier clase de script de JavaScript.

La firma digital de los contenidos de un archivo JAR puede realizarse para cada uno de los archivos o para grupos

Mediante la utilización de la firma digital es posible asegurar que el código que se ejecutará en el ordenador del usuario proviene de una fuente que es fiable.

Utilizando el Compresor JAR

El formato de compresión de archivos JAR fue introducido por Sun Microsystems con la versión 1.1 del *Java Development* Kit. Se trata de un tipo de archivo comprimido que permite introducir todas las clases Java y ficheros JavaScript en un único archivo comprimido para cada página, además, este archivo puede ir firmado digitalmente para asegurar al usuario que no ha sido modificado por fuentes ajenas al origen del archivo. La firma digital de los contenidos de un archivo JAR puede realizarse independientemente para cada uno de los archivos contenidos en un JAR o para grupos de ellos.

Para que el compresor JAR pueda firmar archivos, tendremos que adquirir uno o varios certificados de alguna entidad certificadora e instalar el certificado de firma en nuestra propia copia de Communicator.

Una vez instalado nuestro certificado de firma, seguiremos los siguientes pasos para crear scripts firmados:

- Incluiremos los atributos *ARCHIVE* e *ID* en el primer TAG <SCRIPT> de nuestra página o canal. El resto de los scripts contenidos en el archivo JAR pueden ser identificados por su atributo *ID*.
- En cada script, daremos de alta los privilegios antes de ejecutar cualquier método en modo seguro (osea, con privilegios especiales). Después de la llamada principal deshabilitaremos el modo seguro.
- Finalmente comprimiremos todos nuestros scripts en un único archivo y los firmaremos con el compresor JAR.

Para obtener más información sobre la forma de funcionamiento del compresor/firmador JAR podemos visitar la página WEB de Netscape ***http://developer.netscape.com/software/signedobj/jarpack.html***.

Petición de Privilegios Especiales

Además de ir digitalmente firmado, cierto código debe pedir autorización al usuario utilizando un "objetivo" que representa los recursos del sistema que se necesitan para la acción a realizar.

Como vimos en el primer ejemplo del artículo, la petición de privilegios requiere una única línea de código:

```
netscape.security.PrivilegeManager.enablePrivilege
("objetivo");
```

Una vez que se finaliza la llamada al código privilegiado, Communicator automáticamente le quita los privilegios a nuestro script, pero siempre es bueno indicarlo explícitamente mediante una llamada a *revertPrivilege* (véase de nuevo el código de Flotante.HTM).

El nivel de privilegio al sistema del cliente está representado por los llamados objetivos, que pueden ser especificados como primitivas o macro objetivos. Las primitivas representan acciones privilegiadas concretas, mientras que los macro objetivos representan conjuntos de primitivas. Un ejemplo de macro objetivo es *CanvasAccess*, que permite visualizar texto o gráficos en cualquier parte de la pantalla, sin bordes, botones ni menús.

Acciones que Necesitan Código Firmado

Como hemos visto, podemos crear un canal con extensos contenidos, incluyendo layers con distintos tipos de datos, que no necesitan código firmado. No obstante, ciertas acciones requieren la autorización del usuario, como podrían ser la escritura en su disco duro, la creación de ventanas sin bordes ni menús, añadir o quitar barras de estado a una ventana, crear ventanas independientes, cerrar incondicionalmente ventanas, obtener información de un evento arrastrar y soltar, etc.

Algunas acciones requieren la autorización del usuario: escribir en su disco duro, crear ventanas sin bordes, añadir barras, etc.

Para obtener una lista exhaustiva de las operaciones que necesitan autorización podemos visitar la dirección de Netscape ***http://developer.netscape.com/library/documentation/signedobj/***.

Aunque Communicator escribe en el disco duro cuando mete en el caché los contenidos del canal, o abre una ventana sin bordes para todos los webtops, estas acciones no necesitan estar firmadas, ya que son realizadas por Communicator bajo la autorización de la firma digital de Netscape.

Reglas de Uso de Código Firmado

Hay una serie de normas de obligado cumplimiento para la utilización de código firmado:

- Si un script que haya sido previamente firmado es modificado de alguna manera, tendremos que volver a firmarlo.
- Para cada script que pida privilegios especiales, todos los scripts invocados desde el mismo documento HTML habrán de ser firmados.
- No se puede firmar código que incluya el sintaxis para URLs *javascript:*

Conclusiones

Tras este artículo estamos preparados para desarrollar canales Netcaster que aprovechen toda la potencia que nos ofrece la nueva versión de JavaScript, la 1.2.

Por supuesto, podremos utilizar JavaScript 1.2 para nuestras páginas WEB, pero una gran base instalada de navegadores no será capaz de visualizarlas correctamente (Netscape 2.X y 3.X y TODAS las versiones de Internet Explorer).

Bibliografía

- "HTML Dinámico con Netscape Communicator y JavaScript (Hojas de Estilo)", por Alejandro M. Reyero. Sólo Programadores número 36.
- "HTML Dinámico con Netscape Communicator y JavaScript II (Layers)", por Alejandro M. Reyero. Sólo Programadores número 37.
- "Netscape Netcaster: Iniciación a la Tecnología Push", por Alejandro M. Reyero. Solo Programadores número 38.

Contactar con el autor

Si el lector quiere hacer llegar sus opiniones, preguntas, ejemplos, o, simplemente, decir "hola" al autor, puede dirigirse a través de la dirección:

E-Mail: axl@las.es
WEB: http://xvt.allgames.com

Intranet con Linux (III)

Carlos Álvaro

Para empezar a comprender los mecanismos que permiten la comunicación cliente-servidor en nuestra intranet, es necesario refrescar algunos conceptos, con objeto de utilizar en lo sucesivo los términos adecuados para cada uno.

Así, hablaremos de "socket" en vez de "puerto", o de "daemon" en vez de "TSR" o "programa residente".

Elementos de esta terminología son los siguientes:

• *Background*: conjunto de procesos concurrentes que se ejecutan en un sistema y que no dependen de un terminal o, si lo hacen, dejan el terminal libre para la ejecución de otros procesos. Un ejemplo sería el getty, (agetty en linux); este es un proceso background que chequea continuamente un terminal a la espera de que alguien introduzca su login para iniciar una sesión.

• *Foreground*: procesos no concurrentes dependientes del terminal donde se ejecutan. Son aquellos que el usuario o un programa batch lanza habitualmente y se reconocen porque es necesario que termine un proceso para que comience el siguiente. Un ejemplo claro es el de la introducción de sucesivas órdenes en una sesión normal.

• *Daemon*: proceso *background*, habitualmente no asociado a terminal y, también habitualmente, de duración ilimitada.

• *Socket*: puerto software de entrada/salida a través del cuál se comunican los procesos cliente con los procesos servidor.

• *Host*: máquina de la Internet que proporciona uno o más servicios.

Aunque esto sea válido como regla general para los diferentes sistemas operativos, vamos a centrarnos en los sistemas UNIX y, más concretamente, en Linux.

Todos los servicios TCP/IP son resueltos por *daemons* (a veces llamados demonios) especializados en la prestación de tipos de servicios determinados. Como norma casi general, podríamos decir que existe un proceso *daemon* para cada uno de ellos. Algunos son estándar para cualquiera que sea la plataforma y otros son propios de una u otra distribución de UNIX. Por ejemplo, está ampliamente extendido el uso de un *daemon* llamado "syslogd" que lee y guarda los mensajes de log que aparece en la red por cada uno de los servicios. Estos mensajes llevan implícito un nivel de prioridad.

En una máquina que proporciona servicios a través de TCP/IP, es decir, servicios Internet, esto se puede realizar de dos maneras. La primera sería manteniendo en continua ejecución todos y cada uno de los *daemons*, escuchando sus correspondientes sockets, a la espera de un requerimiento. La segunda, teniendo un único *daemon* servidor que, recibiendo una petición para un determinado servicio, sea capaz de levantar el correspondiente *daemon* para que la atienda.

Los servicios TCP/IP son resueltos por daemons especializados en la prestación de tipos de servicios determinados

Obviamente, el primer método es muy costoso para la gestión de recursos de la máquina, por lo que la inmensa mayoría de los sistemas, empezando por Unix, emplean el segundo método; más eficiente y, por ser centralizado, más sencillo de administrar. Esta es la razón de la existencia de un proceso *daemon* que ha venido a denominarse el "superservidor", esto es, el servidor que está superpuesto a todos los demás.

El Super-server: inetd

El *daemon* inetd comienza su ejecución con el proceso de arranque del sistema operativo y se mantiene activo hasta el proceso de cierre. Se encarga de permanecer a la escucha de los múltiples sockets para el requerimiento de conexiones entrantes. Cuando recibe un requerimiento, lo redirecciona al servidor adecuado. Como ya se ha dicho, el uso de un super-server permite al resto de los servidores activarse sólo cuando se les necesita.

Pero, ¿cuál es el mecanismo de funcionamiento de inetd? El Internet super-server arranca antes que nadie, crea un socket para cada server y enlaza cada uno al puerto correspondiente del servidor correspondiente.

Escucha toda conexión basada en sockets y espera para recibir los datagramas.

Cuando recibe un requerimiento de conexión, devuelve un 'accept". Inmediatamente, crea un socket para la conexión, pero sigue escuchando al socket original por si se producen nuevos requerimientos.

inetd se encarga de iniciar los daemons que responden a las diferentes peticiones de servicio

Realiza las llamadas al sistema para los *forks* (procesos que crean a su vez otros subprocesos independientes llamados procesos *child*), *dups*, (manejadores duplicados de ficheros) y *execs*, (lanzadores de ejecuciones) del servidor apropiado. Con esto, dicho servidor que se acaba de activar obtiene, para el nuevo socket, su stdin (entrada estándar), stdout (salida estándar) y stderr (salida estándar para mensajes de error). El stdin, stdout y stderr le conectan con el proceso cliente.

Los *daemons* que responden a las diferentes peticiones de servicio, en el momento en que se lanza el sistema operativo, arrancan después de inetd. Más concretamente, es inetd el que se encarga de iniciarlos. Algunos de los *daemons* más importantes para servicios Internet son los siguientes:

- *fingerd*: proporciona información sobre usuarios remotos a través del comando "finger". En Linux, este daemon se llama "in.fingerd".

- *ftpd*: es el servidor Internet para el protocolo de transmisión de ficheros entre máquinas.

- *rexecd*: permite la ejecución remota de comandos, mediante sistemas de autentificación basados en la comprobación de usuarios y passwords entre diferentes máquinas.

- *rlogind*: facilita la ejecución de login remoto mediante el programa rlogin. Utiliza mecanismos de verificación y reconocimiento entre máquinas. En Linux, este *daemon* es "in.rlogind".

- rshd: es otro *daemon* para la ejecución remota de comandos. En la mayoría de los sistemas, Linux incluido, se utiliza a través del programa rsh. El *daemon* Linux se llama "in.rshd".

- *talkd*: permite el establecimiento de conversaciones on-line entre usuarios remotos. En Linux, el *daemon* es "in.talkd".

- telnetd: es el *daemon* que permite la conexión directa a un socket en modo de terminal interactivo. Se utiliza principalmente como terminal remoto de otra máquina a través del socket número 23. El nombre Linux del *daemon* es "in.telnetd".

- *tftpd*: es el *trivial file transfer protocol daemon*, que en Linux corresponde a "in.tftpd", y que consiste en un método sencillo de transferencia de ficheros que no requiere de usuario y password en la máquina remota. En principio se trata de un acceso de sólo lectura, aunque los ficheros remotos pueden ser modificados si existen y tienen permiso de escritura. Debido a sus limitaciones, no garantiza la integridad de los datos cuando se realizan transferencias múltiples usando los comandos "get" y "put". Es un servicio muy cómodo de utilizar, pero el permitir que esté activo no forma parte de una buena política de seguridad.

Configuración de inetd

En Linux, la manera habitual de lanzar el inetd es a través del script /etc/rc.d/rc.inet2.

En el momento de su inicialización, el super-server inetd extrae la informa-

ción para su configuración del fichero que se le pasa como parámetro en la correspondiente línea de rc.inet2. Si dicho parámetro se omite se tomará, por defecto, el fichero /etc/inetd.conf.

Una línea de configuración típica de este fichero contiene los siguientes campos:

- Nombre de servicio.
- Tipo de socket.
- Protocolo.
- Exclusividad.
- Propietario.
- Servidor.
- Argumentos.

• Nombre de servicio: debe ser un nombre válido de los recogidos en el fichero /etc/services. Este fichero es el que recoge tanto el nombre oficial, como los llamados "friendly names", para los servicios Internet.

• *Tipo de socket:* especifica la manera en que los procesos tratan esa conexión. Los valores permitidos son:
- "stream" (corriente de datos).
- "dgram" (datagrama).
- "raw" (sin reconocimiento de formato).
- "rdm" (reliably delivered message).
- "seqpacket" (sequenced packet).

• *Protocolo:* debe ser un nombre válido de protocolo de los reflejados en el fichero /etc/protocols.

• *Campo de "Exclusividad":* se aplica a los sockets de datagramas y puede tener como valores "wait" o "nowait". (Para sockets orientados a stream, el valor es "nowait" constante). Estos modificadores actúan sobre inetd haciendo que, cuando un servidor se conecta con el cliente, libere el socket para poder recibir posteriores mensajes sobre el mismo, es decir, que no espere al final de la conexión, (nowait), para recibir otra petición de un cliente.

• *Propietario*: contiene el nombre de usuario bajo el cuál se lanza el daemon servidor. De esto dependen los permisos sobre los diferentes recursos del sistema de que dispondrá el proceso.

• *Servidor*: es el pathname del programa ejecutable que debe iniciar inetd cuando un requerimiento llega a ese determinado socket. Si el servicio es proporcionado internamente por inetd, el valor debe ser "internal".

• *Argumentos:* son los nombres de programa, opciones, modificadores y parámetros del proceso a ser lanzado. Al igual que en la entrada anterior, si el servicio es proporcionado internamente por inetd, el valor debe ser "internal".

El daemon de Routing de red: Routed

Se conoce como *routing* el mecanismo por el cual las peticiones de los clientes, o las respuestas de los servidores, viajan atravesando las diferentes máquinas de la Internet siendo encaminadas por estas máquinas intermedias hasta su destino.

Este mecanismo de routing puede ser de dos formas: mediante rutas estáticas, donde cada máquina sabe de manera fija hacia qué máquina debe dirigir un paquete que busca una determinada red, o de forma dinámica, donde hay un procesos que se comunica con sus homólogos de las otras redes y mantienen actualizadas una serie de tablas con información sobre las rutas.

Lo habitual es una mezcla de ambas cosas. Hay direcciones que se desean mantener de forma estática por diferentes motivos, bien por obligar a todos los paquetes a atravesar unas rutas donde hay una serie de filtros y de medidas de seguridad, bien por elegir rutas sobre líneas más baratas aunque menos eficientes.

Para las rutas que se establecen de forma dinámica, hay un *daemon* llamado "routed" que lee, maneja, gestiona e intercambia las tablas de routing de los sistemas Unix y muchos otros que no lo son.

¿Cuál es el funcionamiento de routed, el *network routing daemon*? Como ya hemos dicho, el *daemon* de *routing* de red maneja las IRT (*Internet Routing Tables*). Esto se hace utilizando alguna variante del Xerox NS *Routing Information Protocol*, que es el estándar Internet.

El *daemon routed* escucha el socket udp 520 a la espera de recibir paquetes de información de *routing*. Si el host es un *router* entre redes, actualiza periódicamente sus tablas, cotejándolas con las de los *routers* de las redes a las que está directamente conectado, ignorando, por supuesto, el *loopback*, es decir, la red que toda máquina forma consigo misma.

El proceso *routed* detecta si existen varios interfaces de red y asume que tiene que intercambiar paquetes entre esas redes, a menos que los tenga configurados como *"ignored interfaces"*.

El daemon de routing de red maneja las IRT

Cuando recibe un *REQUEST packet*, formula una respuesta (*reply*) basándose en la información mantenida en sus tablas internas. La respuesta generada contiene una lista de rutas conocidas marcadas con un *hop count*, que si es igual o mayor que 16 se considera infinito. La métrica asociada a cada ruta es una métrica relativa al emisor.

La respuesta recibida actualiza la tablas internas si se cumple alguna de las siguientes condiciones:

1) No existe entrada en la tabla de *routing* para una red o host destino, y el *hop count* indica *reachable*, es decir, es menor que 16.

2) El paquete recibido no provenía originalmente de sí mismo, es decir, no le está llegando un paquete generado por él mismo después de haber hecho un recorrido por la red.

3) La ruta recibida es igual de costosa que una entrada de la tabla ya existente, pero que no ha sido actualizada en los últimos noventa segundos.

4) La ruta es menos costosa en métrica.

Cuando se aplica una actualización, *routed* registra el cambio en sus tablas internas y genera un paquete de respuesta reflejando esos cambios, dirigido a todas las redes y hosts conectados directamente a la máquina.

Después, espera un corto periodo de tiempo (treinta segundos) antes de actualizar la *Kernel Routing Table*, para prevenir cualquier situación que pudiera resultar inestable.

Además, chequea periódicamente las tablas. Si una entrada no ha sido actualizada durante tres minutos, pone la métrica a infinito. No la borra hasta pasados 60 segundos para asegurarse de que la invalidación ha sido propagada a la internet local.

Un host actuando como *router internetwork* envía gratuitamente sus tablas, esto es, sin que medie ninguna petición, a todas las redes y hosts conectados a él directamente cada 30 segundos. Esto lo hace utilizando la función de *broadcast*, siempre que ésta exista. La recepción de respuesta de cada red es utilizada para verificar que las redes e interfaces funcionan correctamente. Si no es así, el *router* quita esa ruta de su tabla, sustituyéndola por otra. Si no existe ninguna otra ruta válida, simplemente la borra, (*route drop*).

Otro daemon de Routing: gated

El *daemon gated* es más avanzado y posterior a *routed*, pudiendo reemplazar a éste para manejar protocolos de *routing* múltiples. También reemplaza a los *daemons* de *routing* que hablan el HELLO, el EGP o el BGP, porque él es capaz de entenderse con todos estos protocolos.

Aunque cada vez aparece con mayor asiduidad, *gated* aún no está incluido con todas las distribuciones de UNIX.

Algunas Herramientas Útiles

Llegados a este punto, aunque no hemos profundizado excesivamente en las complejidades del funcionamiento de todos los componentes de nuestro plan de servicios Internet, tenemos los conceptos suficientes para, por lo menos, poderle seguir la pista a un problema que surgiese, con algún servicio de nuestro esquema TCP/IP.

Para podernos meter de lleno a configurar cada uno de los servicios que vamos a colocar en la intranet, faltaría tan sólo hacer referencia a algunas herramientas básicas de gestión de nuestra red. Las principales podrían ser las siguientes:

• hostname: establece el nombre de máquina y de dominio.

• ifconfig: asocia interfaces de red, tales como el loopback, las tarjetas ethernet o las líneas serie, a direcciones.

• netstat: es la gran herramienta para la gestión del estado de la red. Es capaz de asociar la dirección a los bloques de control de protocolos, lo que resulta muy útil a la hora del debugging. Dependiendo de su implementación según la plataforma, da diferentes informaciones relativas al funcionamiento de la red, estas informaciones deben incluir, como mínimo, la visualización de los siguientes puntos:

- Estado de los sockets cliente.
- Estado de las interfaces de red.
- Memoria usada por la red.
- Direcciones de red.
- Estadísticas.
- Tablas de routing.

Las diferentes opciones que modifican el comportamiento de cada una de estas herramientas, se pueden encontrar con visores de ayuda, como el "man" de UNIX.

• arp: gestiona la *Internet-to-Ethernet Address Translation Table*. Se implementa mediante un *stream driver* para la negociación del protocolo, y un *stream module* para la traslación. Es el encargado de avisar cuando descubre una dirección IP duplicada en la red.

• trpt: El *Transliterate Protocol Trace* traduce el buffer de log de traza, que se encuentra con los códigos TCP, a mensajes.

• rarp: es el *Reverse Address Resolution Protocol*. Mantiene tablas contrarias a las de arp, es decir, si arp enlaza la direcciones Internet con las direcciones físicas ethernet, rarp relaciona direcciones ethernet con direcciones IP.

Nuestro primer Servidor

Lo que hemos visto hasta ahora corresponde, en su mayor parte, a los cuatro primero niveles de la pila de protocolos TCP/IP. Centrémonos ahora en el nivel

más cercano al usuario; el quinto nivel o "nivel de aplicación".

Los protocolos de aplicación de la familia TCP/IP han sido escritos especialmente para satisfacer de la manera más efectiva cada uno de los servicios que se pretendían ofrecer. Así, existe uno, o más, protocolos específicos para cada uno de los diferentes servicios. Igualmente, es intrínseco a TCP/IP, la utilización de uno o más sockets diferentes para cada servicio.

Sin embargo, ya veremos que esta relación protocolo-servicio no es biunívoca, ya que existe la posibilidad, y de hecho es común, de utilizar para un servicio un protocolo que no es el suyo. Es de uso común el enviar correo mediante protocolo HTTP o FTP, en vez de por SMTP, o recibir ficheros por HTTP en vez de por FTP. Aunque el caso más frecuente es el de la transmisión de ficheros asociados a correo, es decir, utilizar SMTP asociado a alguna técnica del estilo de MIME, para prescindir de FTP. Esto se hace por razones de comodidad, no de efectividad. Posteriormente, se verán estos casos con algo más de detalle.

Uno de los servicios más recientes que se ofrecen a través de la Internet, y que ha constituido un verdadero éxito en cuanto a la aceptación por parte de los usuarios de la Red, es el de transmisión de páginas de hipertexto.

Pero, ¿qué significa esta palabra? Quizás la mejor definición de "hipertexto" sea "todo aquello que es capaz de ser almacenado como fichero en una unidad de almacenamiento". De esta manera, nos encontramos con que esta palabra abarca un sinfín de posibilidades; desde texto normal y corriente, hasta vídeo y sonidos, pasando por imágenes, animaciones y ejecuciones de programas.

La creación de páginas se realiza utilizando el lenguaje HTML (*HyperText Markup Language*). Se trata de un lenguaje de programación no procedural, es decir, carente de estructuras de control de flujo, que incorpora las características necesarias para manejar ficheros en entorno gráfico.

Cliente, servicio, Servidor y Protocolo

Estos cuatro conceptos son fundamentales a la hora de definir las operaciones que realizamos en un entorno de red intranet/Internet. Veámoslos, uno por uno, para el caso de la transmisión de hipertexto.

El cliente universal capaz de explotar estas características es el *browser* o visualizador, que algunos conocen con los términos navegador o explorador debido a la influencia de dos de las principales casas fabricantes de browsers del mercado.

El servicio que proporciona páginas de hipertexto al browser de la máquina cliente es el de World Wide Web, abreviado, el servicio de Web o de WWW. Como todos los servicios Internet, requiere de uno o más procesos *daemon*, que se ejecutan en el servidor y que responden a las peticiones de los programas cliente.

En el nivel de aplicación, el protocolo TCP/IP encargado de servir estas páginas de hipertexto es el *HyperText Transfer Protocol* (HTTP) y el *daemon* capaz de responder al protocolo HTTP es el *HyperText Transfer Protocol Daemon* (httpd).

Habitualmente, el servidor httpd es invocado por el superserver inetd, aunque también puede hacerse que esté siempre activo como *daemon* independiente.

Nuestra versión de Linux utiliza por defecto el servidor de Web Apache, uno de los más comunes y completos que existen en este momento dentro de la Internet.

Configuración del Daemon httpd de Apache

El proceso httpd se lanza en el momento en que el kernel de Linux pasa a init 3 gracias al script /etc/rc.d/rc.httpd, que llama al ejecutable httpd que se encuentra en /usr/sbin. (El nivel de init habitual, por defecto, para el modo multiusuario es el 2. Muchas versiones y/o distribuciones de Linux utilizan el 3.)

Hay una serie de ficheros de configuración que se leen al iniciarse el servicio, y que son los siguientes:

- **/var/lib/httpd/conf/access.conf:** sirve para configurar los parámetros generales de acceso al servicio, esto es, permitir o denegar tal acceso y establecer los directorios que se van a utilizar.

- **/var/lib/httpd/conf/httpd.conf:** define los parámetros característicos del servicio htpd, es decir, cosas tales como si el *daemon* va a ser llamado por el *internet daemon* inetd o si va a ejecutarse independientemente, los sockets a utilizar (normalmente el ochenta), el userid con el que debe ejecutarse, el número de procesos httpd a lanzar, etc.

- **/var/lib/httpd/conf/srm.conf**: establece parametros característicos del manejo interno de ficheros para el contenido de las páginas html. Aquí se encuentran datos como las extensiones por defecto para los distintos tipos de fichero (imágenes, texto, audio, vídeo, ...) asociadas al gráfico que sirve de icono para cada uno de ellos.

Nuestra versión de Linux incorpora, junto con sus procesos de instalación, los del servidor HTTPD Apache, por lo que apenas deberemos hacer nada para tener nuestra Web funcionando. Sin embargo, es conveniente recordar los conceptos y la información que acaba-

Tabla 1

```
<html>

<head>
<title>PAGINA PRINCIPAL DE LOS SIMPSON</title>
</head>

<body bgcolor=cyan text=black>
<h1 align=center>Los Simpson</h1>
<img src="homer5a.gif">Homer I <i>"El Sabio".</i><p>
Este es <b>Homer Simpson.</b> Le gustan las hamburguesas y ver la televisi&oa-
cuten.<p>
Es de color amarillo lim&oacuten.<p>
Entre sus principales caracter&iacutesticas se pueden
enumerar las siguientes:
<ul>
<li>Estudioso de lo que le rodea.
<li>Observador de la naturaleza humana.
<li>Cultivado en Artes y Ciencias.
</ul>
Homer tiene un hijo llamado <a href="pagina.html">Bart</a>
que es muy buen chico.<p>
<img src="homer.gif">
<cite>Homer, hombre equilibrado
y calmoso.</cite><p>
<table border=1 width="50%">
<tr>
<th>Bart</th>
<th>hijo</th>
</tr>
<tr>
<th>Lisa</th>
<th>hija</th>
</tr>
<tr>
<th>Maggie</th>
<th>hija</th>
</tr>
</table>
<img align=top src="simpcons.gif">
Padre e hijo trabajan juntos.
<h4 align=center>FIN DE LA PAGINA SIMPSON.</h4>
</body>

</html>
```

mos de ver si pretendemos profundizar en el conocimiento y manejo de este servidor.

El protocolo de transmisión de hipertexto HTTP basa su funcionalidad en la interpretación del *HiperText Markup Language*, HTML. La sintaxis de este lenguaje incorpora todo lo necesario para la creación de páginas Web, por lo que se hacen necesarios unos conocimientos básicos del mismo para todo aquel administrador que pretenda mantener este servicio en su intranet.

Diseño de páginas Web: El lenguaje HTML

Como ya se ha mencionado, HTML es un lenguaje no procedural, es decir, carente de estructuras de control de flujo, tal como sentencias selectivas o repetitivas. Se basa en etiquetas que el lenguaje reconoce y que son, en su mayoría, controladoras y modificadoras de la apariencia del hipertexto. Se puede consultar el índice de los contenidos de la revista para obtener más información sobre este lenguaje (aunque algunos no lo reconocen como tal).

Para codificar HTML necesitamos, como en cualquier otro lenguaje, un editor de textos como vi o emacs. Vamos a escribir, en el directorio /var/lib/httpd/htdocs DE LA MAQUINA SERVIDOR, dos pequeños ficheros HTML de los que nos serviremos como base para una breve explicación de los elementos fundamentales de este lenguaje (véanse las tablas adjuntas). Nuestros ficheros se llamarán "principal.html" y "pagina.html". La explicación se ciñe exclusivamente a su contenido, por lo tanto es muy parcial e incompleta, y no debe ser tomada como base para el aprendizaje de HTML. El diseño no está cuidado, (sobre todo en lo relativo al colorido) y los gráficos utilizados son muy comunes, y pueden ser encontrados en multitud de páginas accesibles a través de la Internet.

Para poner nuestras dos páginas en el servidor, en nuestra distribución de Linux utilizando Apache, el directorio raíz por defecto para los documentos de hipertexto es /var/lib/httpd/htdocs. Aquí es donde crearemos (o colocaremos, si han sido previamente creados en otro lugar) "principal.html" y "pagina.html", así como los gráficos reseñados en el código

Tabla 2

```
<html>

<head>
<title> home page de Bart Simpson </title>
</head>

<body bgcolor=olive text=white link=red vlink=black alink=maroon>
<h1 align=center> Bart Simpson </h1>
<img aling=center src="bart.gif"> Este es Bart.<p>
<p>
Bart es el hijo mayor de <a href="principal.html">Homer Simpson</a>.
Es educado y respetuoso con todo el mundo.<p>
Entre sus caracter&iacutesticas <u>f&iacutesicas</u> se enumeran:
<ol>
<li>Cabellos peinados a rastrillo.
<li>Ojos de rana.
<li>Sonrisa de Picapiedra.
</ol>
<img align=right src="bart-sch.gif">
<img src="bart-und.gif"><p>
</body>

</html>
```

HTML que hemos escrito. También podríamos haber definido otro directorio en vez de utilizar el que viene por defecto. Esto se puede hacer modificando la entrada 'DocumentRoot' en el fichero de configuración de Apache /var/lib/httpd/conf/srm.conf.

Lo que sí deberemos hacer es modificar la entrada 'DirectoryIndex' de /var/lib/httpd/conf/srm.conf que, por defecto, tiene el valor "index.html", y establecer el valor "principal.html". Una vez hecho esto, nuestro servidor de páginas Web está listo para servir el hipertexto que hemos codificado. Sólo nos queda la parte del cliente, es decir, cómo visualizar el resultado de nuestro trabajo.

El cliente HTTP

Del mismo modo que, como ya hemos indicado, cada servicio Internet es generalmente resuelto por un *daemon* especializado, lo mismo podría aplicarse a la parte cliente.

Así, lo normal es encontrarse con un ejecutable "ftp" ("ftp.exe", en el caso de entornos OS/2 o MS-DOS) para la transmisión de ficheros, o un ejecutable "mail" ("mail.exe") para la petición, lectura o envío de correo.

Sin embargo, y últimamente, la mayoría de las casas comerciales de software dedicado a temas de la Internet apuestan por un "cliente universal", es decir, una sola aplicación que pueda, simultáneamente, actuar como cliente de los distintos servicios Internet. Ese es el enfoque que se pretende dar a los browsers de hoy en día.

Para nuestras pruebas vamos a utilizar uno de los mejores (según opinión de muchos, el mejor) que existen disponibles para multiples plataformas, incluida, claro está, Linux: el Netscape Navigator 3.01. También usaremos unos gráficos que se pueden encontrar en multitud de URL's de la Internet. (Por ejemplo, en "http://www.dayna.com/emppages/bodily/home.html" o en "http://molokai.ucsd.edu/council/World/cool.html").

El paquete de archivos para Netscape Navigator 3.01 se puede recoger de "ftp://archive.netscape.com/archive/navigator/3.01/unix/netscape-v301-export.x86-unknown-linux-elf.tar.gz", y debe ser utilizado de acuerdo con los términos de la licencia que viene incluida.

Las casas de software apuestan por un "cliente universal": una sola aplicación que pueda actuar como cliente de los servicios Internet

De todas formas, incluimos el paquete "intra2sup.tar.gz" donde se encuentran ambas cosas; gráficos y browser, y que debe ser instalado de la siguiente forma:

- Nos situamos en el directorio temporal DE LA MAQUINA CLIENTE con "cd /tmp".
- intra2sup.tar.gz puede ser copiado en este directorio y descomprimido desde aquí utilizando "gzip -d intra2sup.tar.gz".
- intra2sup.tar es el archivo empaquetado que queda después de la descompresión, y se desempaqueta mediante "tar xvf intra2sup.tar".
- Ubicaremos los archivos "*gif" resultantes en el directorio raíz de hipertexto DE LA MAQUINA SERVIDOR. Un método para hacer esto, (que no es el mejor) es el siguiente. En un disco formateado EN LA MÁQUINA CLIENTE copiamos los mediante "tar cvf /dev/fd0 *gif", trasladamos el dis-

quete A LA MÁQUINA SERVIDOR, nos situamos en el directorio con "cd /var/lib/httpd/htdocs", y extraemos los ficheros de gráficos con "tar xvf /dev/fd0".

• Volvemos A LA MÁQUINA CLIENTE y creamos el directorio de Netscape con "mkdir /usr/netscape" y nos situamos allí con "cd /usr/netscape".

• Desempaquetamos el .tar de netscape con "tar xvf /tmp/netscape-v301-export.x86-unknown-linux-elf.tar".

En la distribución de Linux que utiliza Apache, el directorio raíz para los documentos de hipertexto es /var/lib/httpd/htdocs.

• En caso de no tener configurado el entorno gráfico, utilizaremos "XF86Setup" para hacerlo y, una vez arrancado con "startx", desde una ventana del shell ejecutaremos "/usr/netscape/netscape".

• En la ventana para la URL escribimos el nombre de la máquina servidor con el formato "http://maquina.dominio" o bien, simplemente, su dirección IP.

• Nuestras páginas de hipertexto, ubicadas en el servidor, deben aparecer visibles en el cliente.

Tabla 3

Las etiquetas <html> y </html> delimitan a cada documento. Un documento HTML consta de dos secciones bien diferenciadas: la sección "head", donde se enmarcan los datos de caracter general relativos al documento completo, y la sección "body" que, como su propio nombre indica, es el cuerpo del documento.

En nuestro primer ejemplo, complementamos la etiqueta <body> con un color de fondo y otro de texto. En el segundo ejemplo, también determinamos otros factores, como cuál es el color para los elementos de enlace sin usar o cuando ya han sido usados.

Los sucesivos títulos que van a aparecer a lo largo de la página se suelen diferenciar con un tamaño de letra especial. Estos tamaños de letra se establecen con la etiqueta <hn>, donde 'n' es un número, del uno al seis, que representa en orden decreciente dicho tamaño.

La posición se determina mediante <align> y un valor; "left", "right", etc.

Los gráficos aplicables a páginas de hipertexto deben ser dados en formato "gif" o "jpg" mediante la etiqueta <img>.

Aparte del tamaño y la posición, también se puede jugar con el tipo de letra. Hemos utilizado como ejemplo las etiquetas <i>, para la letra itálica, y <b>, (bold), para negrita.

Los caracteres del ascii extendido, es decir, los que se encuentran por encima del 127, varían de unas páginas de códigos a otras, y tienen una codificación especial que obedece al patrón "&codigo;". Este es el caso de las vocales acentuadas, ("á", "ò", "û", etc), la letra "ñ", ("ñ"), y muchas otras.

Debido a que HTML no tiene en cuenta los espacios en blanco de los ficheros fuente, el final de un párrafo debe ser especificado utilizando <p>.

Se pueden emplear listas de dos tipos: numeradas, como en "pagina.html" o sin numerar, como en "principal.html". Las etiquetas respectivas son <ol> y <ul> para encuadrar la lista, y <li> para cada uno de los elementos.

Para marcar un elemento, ya sea de texto, gráfico o de otro tipo, como "activo", se usa <a>. Un elemento activo es un enlace a otro elemento: un gráfico, un sonido, otra página HTML o, incluso, una posición distinta dentro de la misma página.

Siempre dependiente de la configuración del browser es la etiqueta <cite>, que presenta un texto con un tipo de letra especial.

Por último, HTML también es capaz de distribuir elementos en tablas a través de <table>.

Conclusión

Nuestro servicio de hipertexto está ya configurado, activo y disponible mediante el servidor Apache para Linux. En posteriores artículos veremos el funcionamiento y configuración de otros, no menos importantes, servicios Internet.

Hay que reseñar que la activación y puesta en marcha de un servicio requiere la presencia de una persona encargada de su gestión y mantenimiento. Este punto que, a veces, no recibe la importancia que se le debe, marca la diferencia entre una intranet que funciona satisfaciendo las exigencias de los clientes, y otra que no.

FidoNet: No es Internet todo lo que reluce

Enrique Díaz

No sólo a través de Internet podemos tener correo electrónico, ficheros y áreas de discusión. FidoNet es una red que en cuanto a correo electrónico no tiene nada que envidiar a Internet; en honor a la verdad hay que decir que el sistema de correo electrónico de FidoNet deja en pañales al de Internet, en cuanto a eficiencia se refiere. Como muestra decir que un servidor ayer mismo se conectó, recibió doscientos mensajes procedentes de nueve áreas distintas y se desconectó... todo ello en menos de un minuto y con un módem de 14.400 baudios.

FidoNet es una red amateur que no cobra por sus servicios y que no está subvencionada

FidoNet es un sistema amateur de correo electrónico y rutado de ficheros. Como tal, todos sus participantes y operadores son voluntarios no pagados. Desde su nacimiento como medio de intercambio de mensajes por parte de unos cuantos amigos, ha ido creciendo hasta llegar a tener unos treinta mil nodos en cuatro continentes.

Lamentablemente, la explosión de Internet ha conseguido ponerla en recesión desde hace algunos años.

Como hemos dicho, FidoNet es una red amateur que no cobra por sus servicios y que no está subvencionada. Por ello todos los programas que la hacen posible están pensados para ahorrar hasta la última peseta de las siempre caras comunicaciones. La conexión entre los nodos no es permanente, sino que existen unas normas para que los nodos se llamen unos a otros, en horarios y turnos establecidos.

Las BBSs

Los nodos de FidoNet son las BBSs. Una BBS es un ordenador que atiende llamadas de usuarios y pone a su disposición ficheros y mensajes. La persona que ha puesto su ordenador, su tiempo y su dinero a tu servicio es el *SysOp* (acrónimo de *System Operator*, operador del sistema) de la BBS.

Para apuntarse a una BBS sólo hace falta un programa de comunicaciones (preferiblemente que soporte códigos ANSI) y llamar a una de ellas (podéis encontrar una lista con BBSs de toda España en la dirección http://www.conexis.es/~marina). En la primera llamada se nos pedirán algunos datos, como el nombre, la dirección y el número de teléfono. Una vez rellenado el cuestionario tendremos acceso a la BBS, normalmente con unos privilegios muy cortitos hasta que el *SysOp* nos conozca algo mejor y nos otorgue mayores

FidoNet es una red mundial amateur de rutado de correo y ficheros. Sus servicios, en ocasiones, dejan 'en pañales' a los de Internet. Y todo gratis. En este artículo veremos la organización de la red, su funcionamiento y cómo hacerse miembro de esta gran red.

privilegios. Esto quiere decir que en principio tendremos el tiempo de conexión limitado a unos quince ó treinta minutos diarios y tendremos que cumplir con unos *ratios* para llevarnos ficheros, es decir, si queremos llevarnos algún fichero de la BBS deberemos dejar algún otro a cambio.

Habitualmente el *ratio* es de 1 a 4; por cada Kilobyte que dejemos en la BBS podremos llevarnos otros cuatro. Al principio también es posible que el acceso a las áreas de mensajes esté limitado a las áreas locales de la BBS, esto es, las que no se envían a otros sistemas para su distribución. Cuando el *SysOp* nos conozca algo mejor, nuestro tiempo de conexión aumentará, no tendremos que cumplir *ratios* y tendremos acceso a todas las áreas: locales, nacionales e internacionales.

Por cada Kilobyte que dejamos en la BBS podemos llevarnos otros cuatro

Aunque al principio todos nos apuntamos a una BBS para tener una fuente gratuita de *software,* lo que nos enganchará finalmente será la mensajería. Los primeros mensajes los pondremos en modo terminal, pero lo mejor es que nos hagamos lo antes posible con un programa lector de correo *off-line,* como por ejemplo *BlueWave.* Para utilizarlo especificaremos en la propia BBS las áreas de nuestro interés. Un programa de la BBS mantendrá un sistema de punteros que distinguirá los mensajes que hemos leído y los que no. Cuando llamemos a la BBS, el programa buscará los mensajes nuevos en las áreas que tenemos marcadas, los comprimirá (con el compresor que hayamos elegido) y nos lo enviará.

En este punto ya podemos colgar y activar nuestro programa local, que descomprimirá y agrupará los mensajes por áreas. De este modo los leemos tranquilamente y contestamos de igual modo.

Para *subir* (mandar) los mensajes el proceso será a la inversa, comprimimos las respuestas, nos conectamos y se lo enviamos a la BBS, que descomprimirá el paquete y repartirá cada mensaje en su área correspondiente. ¿Por qué comprimir los mensajes? Pues para que nos ahorremos tiempo de conexión. La filosofía de minimizar la factura de teléfono también se extiende a los usuarios de las BBS. Todo el proceso de enviar nuestras respuestas y recoger el correo nuevo no nos llevará más de dos o tres minutos. Si esto aún no os parece mucho existe un medio todavía más rápido: hacerse punto, pero lo veremos más adelante.

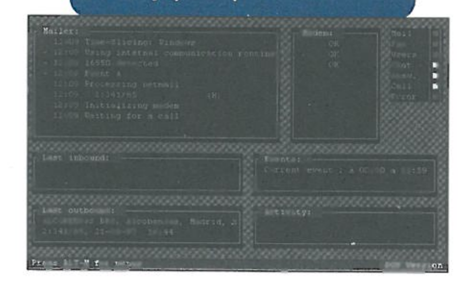
Los mensajes

No hace falta ser Camilo José Cela para escribir un mensaje. Lo cierto es que la mensajería tiene su propia gramática y semántica, y el hecho de ser unos expertos en el manejo del lenguaje no nos garantizará el éxito en la comunicación. Lo fundamental es que se entienda y escribirlo rápido, que no es cuestión de pasarse las horas muertas redactando un mensaje. Muchas veces basta con separar frases y párrafos para que se entienda perfectamente, y se puede prescindir de otros signos de puntuación. De hecho se considera incorrecto *afear* al prójimo con faltas de hortografía o palarbas tarbucadas como aaquí ocurre.

Lo que sí debemos tener en cuenta es el medio. Los mensajes se leen en pantalla, que cansa mucho más la vista que si leemos en papel. Un párrafo kilométrico dificulta la lectura del mensaje en pantalla, por lo que facilitaremos su lectura si empleamos párrafos cortos separados por un retorno de carro. Aunque el texto no requiera abrir un nuevo párrafo, los ojos del lector probablemente sí lo requieren.

Otro aspecto a tener en cuenta en los mensajes es el 'quote' o acotaciones.

Figura 1. El mailer automatiza la entrega y recepción de correo.



Las acotaciones son fragmentos del mensaje que estamos respondiendo y que se mantienen a modo de referencia para que el destinatario sepa exactamente a qué le estamos respondiendo, y para que el diálogo electrónico se parezca lo más posible a un diálogo 'normal'.

Por ejemplo, si yo le escribo a Jesús un mensaje tal que así:

Hola Jesús
Que tal el trabajo? Supongo que mejor,
porque te han puesto un ordenador más potente, no?

Él me lo podría acotar de la siguiente manera:

ED>Hola Jesús
Hola
ED>Que tal el trabajo?
Tirandillo...
ED>Supongo que mejor,
ED>porque te han puesto un ordenador más potente, no?
No me hables, este trasto no vale para nada :(

Como se puede ver, es un buen método para mantener el hilo de la conversación, sobre todo si estamos escribiendo en alguna área pública, ya que el resto de los usuarios puede captar rápidamente la conversación.

Volvemos ahora al carácter amateur de FidoNet para evitar el abuso de las acotaciones. No se debe acotar un mensaje larguísimo sólo para poner al final "Estoy de acuerdo". Aparte de no resultar claro, obligamos a nuestro *SysOp* a transmitir un montón de bytes innecesarios, y algunos montones de bytes innecesarios multiplicados por todos los usuarios de FidoNet,

pueden dar como resultado algunos *megas* que se rutan innecesariamente.

El SysOp pondrá a tu servicio su ordenador, su tiempo y su dinero

En este sentido, hay partes de un mensaje que no se deben acotar:

- Las cabeceras automáticas, como por ejemplo "El 15/10/97 a las 15:38 te dirigiste a mí en éstos términos:"

- Los *origins*, unas frases más o menos ingeniosas que se ponen al final del mensaje. Por ejemplo: "Dar aquí con un martillo para tener un monitor nuevo."

- Las firmas.

- La información adicional del mensaje: el path y el seen-by, que muestran la ruta seguida por el mensaje hasta nosotros y quién lo ha visto.

Normas de cortesía

Ante todo hay que advertir que el ambiente que reina en FidoNet es totalmente diferente al de Internet. Su carácter amateur le dota un espíritu de colaboración y de ayuda entre todos. Sin embargo, FidoNet es lo suficientemente grande como para caer por su propio peso, y debido a ello se hace imprescindible seguir algunas normas básicas. En Internet puedes hacer lo quieras mientras pagues tus facturas y no sea delito. En FidoNet hay que seguir algunas normas de obligado cumplimiento:

- No utilizar la red para fines ilegales. Se refiere básicamente a la piratería.

- Está prohibido insultar o mantener actitudes racistas, sexistas o discriminatorias hacia cualquier grupo étnico o religioso. A este respecto hubo una anécdota curiosa:

Los chicos y las chicas mantenían una 'guerra' de chistes machistas y feministas en el área de chistes. Debido a que hay muy pocas chicas en la red (esto es así en FidoNet e Internet y nadie sabe por qué) los chistes machistas eran la inmensa mayoría. Alguien, amparándose en esta norma solicitó que se prohibiera este tipo de chistes en el área. Inmediatamente salieron a la palestra todas las chicas para defender la continuidad de esta particular 'guerra', y los chistes continuaron. El moderador del área ni siquiera tuvo que intervenir. Esto nos puede servir para ilustrar que el sentido común es superior a las normas.

- No utilizar palabras malsonantes. Esta norma es 'muy americana' (FidoNet nació en Estados Unidos). En España la hemos resuelto elegantemente utilizando algo de picardía. Por ejemplo podemos decir: "... pues a mí ese virus me ha hecho una put*da enorme..." (como todos sabemos en informática el asterisco sustituye a cualquier carácter o cadena de caracteres (si lees algún taco, seguramente has pensado mal).

No es necesario ser Camilo José Cela para escribir un mensaje; éstos tienen sus propias reglas

Además de estas normas, existen otras que se consideran 'de buena educación'. Por ejemplo ESTÁ MAL VISTO ESCRIBIR EN MAYÚSCULAS, ES COMO SI GRITÁRAMOS. Las mayúsculas se emplean también para resaltar palabras IMPORTANTES, aunque eso mismo podemos hacerlo utilizando *asteriscos*.

También debemos ceñirnos al tema del área en la que estamos escribiendo. Si yo estoy suscrito a un área de OS/2, probablemente no me hará gracia recibir mensajes relativos a otros sistemas operativos que no utilizo ni me interesan, aunque si tenemos una pregunta para la que no encontramos un área adecuada, se puede preguntar en otra área parecida. Por ejemplo, si en el área de OS/2 advertimos a los demás que no sabemos dónde preguntar y hacemos una consulta sobre algún tema relativo a Windows NT, nos contestarán si lo saben, y en todo caso seguramente alguien nos remitirá al área correcta.

Tabla 1. Emoticonos.

:-)	Sonrisa
X-D	Partiéndose de risa
:-(	Triste
;-)	Guiño
:-/	Mosqueado
}:-)	Pícaro, irónico, pillín
:-l	Serio
O:-)	Santo, bondadoso
:-?	Duda, pregunta.
:'-(	Llorando
:-D	Riendo
:'-)	Llorando de felicidad
8-)	Con gafas
:-)'	Se cae la baba
:*)	Beodo
:-*	Beso

Todas las recomendaciones que han sido comentadas y las normas que se deben cumplir en Fidonet, se pueden resumir en dos actitudes que es preciso mantener en todo momento:

1. No molestes a los demás.
2. No te molestes fácilmente.

Las emociones

En un medio escrito como es la mensajería electrónica, se hace difícil expresar emociones o actitudes, lo cual puede dar lugar a malentendidos. Para expresar emociones podemos hacer uso de las llamadas caritas, caretos o emoticonos, que sirven para expresar emociones. Los emoticonos representan caras. En el emoticono :-) los dos puntos son los ojos, el guión la nariz y el cierre de paréntesis la boca. Si aún no lo véis podéis girar el papel 90 grados a la derecha. En la tabla 1 tenéis una lista muy básica de emoticonos.

Si, por ejemplo, en un mensaje alguien comenta que le gusta el Linux y le contestamos así:

>Pues a mí el Linux me encanta...
Eso es porque no tienes ni idea de informática.

Puede ser que el amante de Linux se sienta ofendido. En cambio la cosa cambia radicalmente si en lugar de eso decimos:

>Pues a mí el Linux me encanta...
Eso es porque no tienes ni idea de informática };-).

Con el emoticono de pícaro le hemos dado un giro irónico a la frase. Queremos decir justamente lo contrario de lo que decimos.

La estructura de FidoNet

FidoNet nació en el año 1983 gracias a que a un tal Tom Jenning se le ocurrió crear un programa de comunicaciones que le permitiera comunicarse con sus amigos de forma cómoda. Para conseguirlo, construyó un ordenador con distintas piezas de otros al que decidió llamar Fido, y desarrolló un nuevo programa, de nombre BBS.EXE (hay quien afirma que Fido no era más que el perro de Jennigs; y, de hecho, la mascota de FidoNet es un perrito).

Poco imaginaba Jenning el impacto que iba a tener su invento "casero". A sus amigos se fueron uniendo los amigos de los amigos, y al cabo de algo más de un año ya existían más de mil nodos en Estados Unidos. Tan sólo un año más tarde esa cifra ya se había duplicado. Este rápido e inesperado crecimiento obligó a diseñar sistemas de rutado de correo al mínimo coste y también a dotar a la red, que ya por entonces se llamaba FidoNet, de una cierta estructura básica. Esta estructura permitió que en muy poco tiempo alcanzara una dimensión internacional.

De este modo FidoNet se divide en zonas, éstas a su vez en regiones, las regiones en *networks* que están formadas por nodos (esto es, las BBSs en sí). Las BBSs a su vez pueden tener puntos (es decir, usuarios cualificados), que constituyen el nivel más básico de la organización de FidoNet. Las zonas se corresponden aproximadamente con los continentes, las regiones con países y las *networks* con provincias, áreas geográficas o autonomías. En la actualidad, existen cuatro zonas fundamentales en el mundo: América del Norte, América del Sur, Europa y Oceanía.

Para poder hacer llegar un mensaje a cualquier usuario del mundo se ha ideado un sistema de direcciones numérico; es poco personal pero desde luego muy efectivo. Por ejemplo, la dirección de FidoNet del que escribe estas líneas es 2:341/85.17. Esta dirección identifica la zona 2 (Europa), región 34 (España), *network* 1 (Castilla-Centro), nodo 85 (la BBS) y punto 17 (un servidor).

Además de esta estructura organizadora existe otra jerárquica para garantizar el cumplimento de las normas de FidoNet. Esta otra estructura la componen los nodos coordinadores de cada uno de los niveles. De este modo, existe un coordinador de puntos, de network, de región, etc. Si alguien incumple insistentemente las normas de FidoNet, los coordinadores pueden amonestarlo, o excomunicarlo (dejarle fuera de la red) como última medida.

Las áreas de mensajes

Las áreas de mensajes también tienen su propia estructura que reproducen más o menos la estructura de FidoNet. De este modo existen los siguientes tipos de áreas:

- Locales. Son las propias de una BBS. No se rutan al resto de la red, de modo que sólo son accesibles para los usuarios de esa BBS.

- Nacionales. Aquí hay que distinguir dos tipos; las que se rutan a todo el país o sólo a una *network* (como por ejemplo la de compra-venta de la 341, que sólo se distribuye entre los nodos de Castilla-Centro).

- Internacionales. Se distribuyen por todo FidoNet, de modo que participan usuarios de todo el mundo.

El carácter Amateur de FidoNet dota a la red de un espíritu de ayuda y colaboración entre todos

Las áreas de mensajes están moderadas por un responsable del área y tienen sus propias reglas, que de normal son bastante sencillas. En las normas se especifica cuál es la temática, los idiomas permitidos y cómo se resuelven las even-

Figura 2. Las acotaciones nos ayudarán a seguir el hilo de la conversación.

tuales disputas entre usuarios. Antes hemos comentado que las normas de FidoNet obligan a una cierta cortesía, pero si 'te va la marcha' puedes acudir al área de insultos y despropósitos. Como se puede ver, cada cosa en su lugar.

Cómo hacerse Punto

El punto es un usuario cualificado de FidoNet y es el mayor grado de integración que se puede alcanzar en la red (de hecho un punto viene a ser una BBS que no atiende llamadas y sólo ruta su propio correo). Esta mayor integración permite una serie de ventajas sobre el resto de los usuarios, como poder pedir ficheros a cualquier BBS del mundo sin dejar nada a cambio, pero la mayor ventaja está en la mensajería.

Un punto selecciona las áreas de mensajes de su interés y, según van llegando los mensajes a las diferentes áreas, un programa de la BBS añade los mensajes a un paquete comprimido. El punto llamará más tarde a la BBS con un programa especial para recoger su correo comprimido y enviar los mensajes salientes de igual manera. De este modo el tiempo de conexión semanal a la BBS no irá más allá de los diez minutos. A cambio nosotros tan sólo debemos respetar las normas de FidoNet y, si es posible, tener espíritu de colaboración, ayudar en lo posible al resto de los usuarios... en definitiva, participar del espíritu de FidoNet.

Para ser punto necesitaremos un periodo previo de rodaje como usuario terminal utilizando programas lectores de correo off-line (que es casi tanto como ser punto). Una vez tengamos unos conocimientos mínimos sobre la red y su funcionamiento, será hora de que le pidamos a nuestro *SysOp* que nos haga punto. A estas alturas ya nos habremos escrito unas cuantas veces con el *SysOp* para pedirle ayuda, de modo que si considera que somos gente de bien nos dará un número de punto sin problemas.

En cuanto a la parte técnica, necesitaremos tres programas: un *mailer*, *tosser* y un lector.

Un *mailer* es un programa que automatiza la gestión del correo y son usados por los propios *SysOps* para intercambiar el correo con otras BBS's. En el caso de un punto el *mailer* llamará a la BBS, presentara nuestro nombre y contraseña, enviará nuestro correo de salida y recogerá el nuevo correo que tengamos acumulado en la BBS. Incluso es posible programar el *mailer* para que llame de forma desatendida a determinadas horas.

El *tosser* se encarga de desempaquetar el correo y poner cada mensaje en su área correspondiente, así como el empaquetado del correo saliente.

Finalmente el programa lector nos permitirá leer y contestar mensajes, dándoles el formato adecuado.

También existen algunos programas 'todo en uno', es decir, son *mailer*, *tosser* y lector, con lo cual nos podemos evitar el engorro de hacer que tres programas colaboren juntos (incluso alguno más si contamos el programa compresor o deseamos usar un editor de textos distinto al que trae el programa).

Montar tu propia BBS

Montar una BBS es un proyecto siempre apasionante, y para ello no se necesita mucho; sólo ese viejo ordenador que tienes apartado, un módem y ... tiempo, mucho tiempo, muchas pruebas y muchos quebraderos de cabeza. Si utilizas un sistema operativo con multitarea real como OS/2 o Linux no hará falta tener un segundo ordenador, ya que se puede ejecutar el software de BBS en segundo plano sin problemas. Para montar una BBS a la que llaman unos cuantos amigos es todo lo que se necesita. También resulta muy recomendable hacerse amigo de un *SysOp* y tener alguna experiencia como punto de FidoNet. Si ya eres punto de una BBS sólo tendrás que añadir un programa de BBS a tu software de punto. En cualquier BBS encontrarás este software, que suele ser gratuito siempre que no se utilice para fines comerciales.

Ser punto representa el grado más alto de integración en la red

Integrar la BBS en FidoNet ya es otro cantar, ya que formarás parte de una gran red. Si se quiere montar una BBS mínimamente decente ya no valdrá cualquier ordenador. Al espacio requerido en disco para los ficheros que ofrecerás a los usuarios hay que sumar el espacio nada despreciable que ocuparán las áreas de mensajes. También es casi imprescindible tener una línea de teléfono dedicada. Aunque las BBS se pueden poner con horario limitado (por ejemplo de 23 a 8), en la práctica resulta inviable ya que siempre habrá alguien que no respete el horario. El teléfono sonará a todas horas y al descolgar oirás el pitido de un módem, tu familia empezará a cansarse y... cerrarás la BBS.

También es necesario tener mucho tiempo y muchas ganas. Montar una BBS no consiste sólo en la parte técnica y esperar a que llamen. Una BBS requiere un mantenimiento diario. Pregúntale a cualquier *SysOp* antes de animarte a montar una, te contará lo que no te imaginas.

En cuanto al coste, todo está diseñado para que las comunicaciones sean muy eficientes, de modo que tu factura telefónica no irá más allá de las cinco o seis mil pesetas.

Si todo esto no te ha desanimado ¡Adelante! En FidoNet encontrarás montones de gente dispuesta a ayudarte.

REDES LOCALES

Quién es quién en Windows NT

Enrique Díaz

Abordamos en este artículo la administración de usuarios bajo Windows NT y realizamos un primer acercamiento al sistema de seguridad del sistema operativo. Seremos capaces de controlar lo que puede y no puede hacer cada usuario; algo que debemos tener muy claro a la hora de montar una futura intranet.

Los objetos en Windows NT, como en todos los modernos sistemas operativos de red, se protegen más o menos solos. Para poder acceder a los recursos y objetos de un dominio es necesario tener una cuenta de usuario. Todas las cuentas de usuario están identificadas por un identificador (SID, identificador único de seguridad), que es un número generado por Windows NT cuando se crea una cuenta. Cuando un administrador concede privilegios a un usuario para acceder a un objeto —como por ejemplo una impresora o un directorio— lo que está haciendo es añadir el número de identificación de la cuenta a la lista de control de acceso (ACL, *Access Control List*) asociada al objeto. Esto es lo que se llama *control de acceso discrecional,* y es lo que permite, por ejemplo, establecer distintos permisos para diferentes archivos de un mismo directorio. De este modo las cuentas de usuario y los permisos sobre los recursos nos permitirán establecer las restricciones para cada usuario.

Para administrar la seguridad de Windows NT disponemos de cuatro tipos de cuentas:

- Usuarios globales.
- Usuarios locales.
- Grupos globales.
- Grupos locales.

Las cuentas de usuario global se crean en un servidor NT y pueden ser usadas tanto en el dominio en que se creó como en los dominios que confían en ese dominio. Es posible establecer privilegios en este tipo de cuenta, pero lo usual es hacerlo en las cuentas de grupo. De este modo para cambiar los privilegios de un usuario basta con cambiarle de grupo, o modificar los privilegios de varios usuarios modificando una sola cuenta de grupo.

Para acceder a los recursos y objetos de un dominio es necesario tener una cuenta de usuario

Las cuentas de usuario local no pueden ser utilizadas fuera del dominio en que se crearon, aunque sí pueden ser añadidas a grupos locales y globales y es posible asignarles derechos. Son parecidas a las cuentas de usuario normales, con la mencionada excepción de que no pueden ser utilizadas fuera de su dominio. Un ejemplo de estas cuentas son las procedentes de un dominio en el que no se confía, de IBM LAN Server o Novell Netware.

Los grupos globales sólo pueden tener cuentas de usuario y tienen la característica de poder recibir derechos de

múltiples dominios. No puede contener otras cuentas de grupo, sin embargo, las cuentas de grupo global sí pueden ser añadidas a grupos de su dominio o de otros dominios en los que se confía.

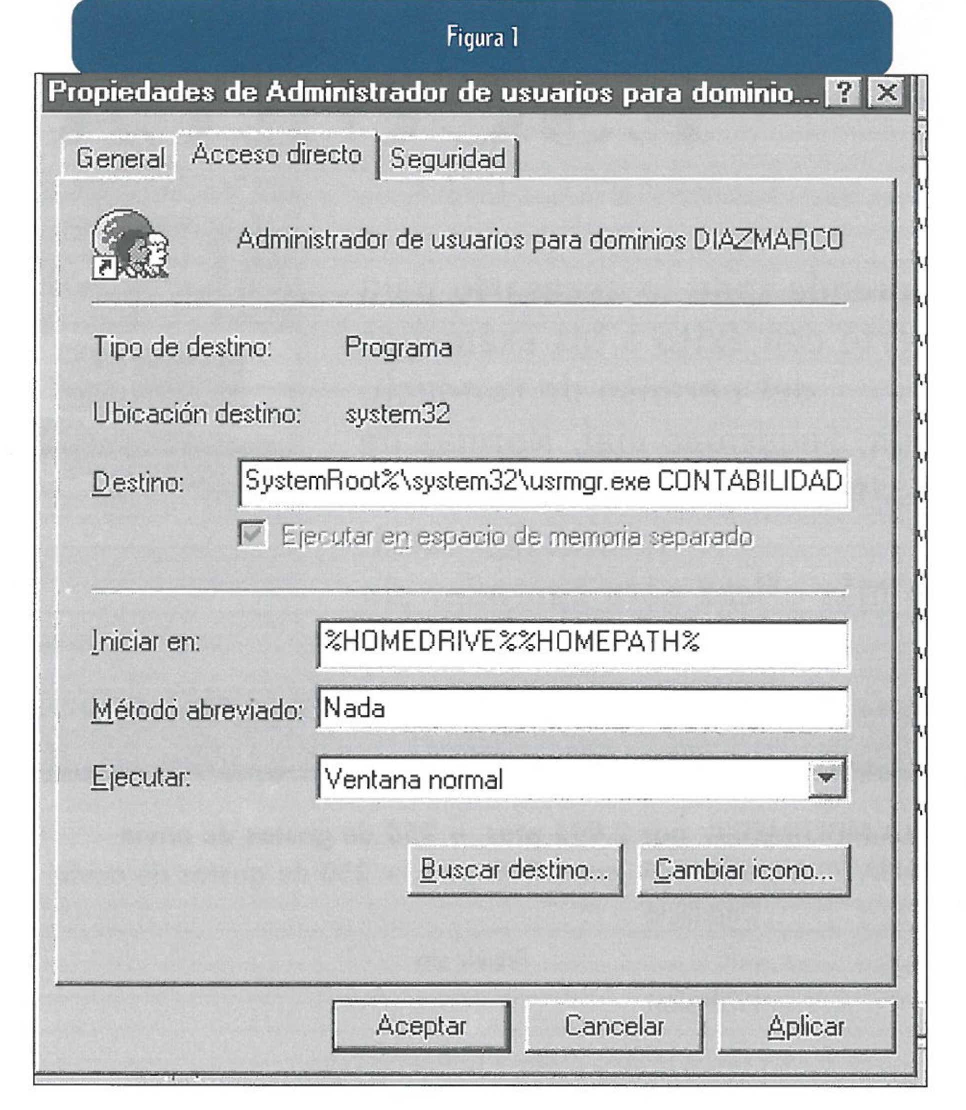
También es posible asignar privilegios a este tipo de cuenta, pero es mucho mejor aprovechar la posibilidad de agregarla a grupos locales. De este modo asignamos los privilegios a los grupos locales y utilizamos el grupo global como método para añadir usuarios a otros grupos locales.

Las cuentas de grupo local pueden contener usuarios y grupos globales, pero no otras cuentas de grupo local. Sólo se le pueden asignar permisos del dominio al que pertenecen.

Sólo corremos un peligro con la cuenta del administrador: olvidar la contraseña

Todos estos tipos de cuenta, utilizados sabiamente, nos pueden simplificar bastante el trabajo. Por ejemplo, podemos utilizar los grupos globales para añadir usuarios a grupos locales de dominios en los que se confía, de esta manera conseguimos extender sus derechos a otros dominios en los que se confía. Por supuesto, también funciona a la inversa; si añadimos grupos globales de otros dominios a nuestros grupos locales, les concedemos privilegios sobre nuestro dominio.

Figura 1



Cuentas de Usuario Predefinidas

Durante la instalación se crean las cuentas de usuario del administrador y del invitado del dominio. La cuenta del administrador es una cuenta especial. Es la que permite la administración del servidor en que se creó y no se puede borrar, deshabilitar ni bloquear. Puede ser renombrada y su contraseña se puede cambiar, pero no caduca. De esta manera evitamos bloquear o eliminar la cuenta accidentalmente si realizamos alguna operación con el conjunto de las cuentas de usuario. Así pues el único peligro que nos queda es perder la contraseña, por ésta y otras razones de seguridad debemos guardarla cuidadosamente. También es aconsejable crear al menos una segunda cuenta de administrador y dejar la creada durante la instalación para un apuro. El administrador es miembro de los grupos: local de administradores (no puede eliminarse de este grupo), global de administradores de dominio y de usuarios del dominio.

Los permisos son diferentes de los derechos; en caso de conflicto prevalecen los derechos

La cuenta de invitado del dominio se crea desactivada, pero puede reactivarse. No puede ser borrada, aunque sí renombrada. Se utiliza para aquellos usuarios que no tienen cuenta en el equipo, en el dominio, o pertenecen a un dominio en el que no se confía. La cuenta de invitado es miembro de los grupos local y global de

invitados y puede ser configurada para que permita el acceso desde la red, en el propio equipo, o ambos.

Cuentas de grupo Predefinidas

Al mismo tiempo que las cuentas de usuario, durante la instalación de Windows NT se crean algunos grupos que responden a unos perfiles de usuario más o menos estándares. Esto nos permitirá asignar unos privilegios genéricos a los nuevos usuarios con tan sólo agregarlos a las cuentas de grupo que nos interese:

- Administradores. Es una cuenta de grupo local que se crea en servidores y estaciones de trabajo Windows NT. Tienen control total sobre el equipo o dominio en el que residen sus cuentas. Pueden añadir estaciones al dominio, asignar derechos de usuario, crear y administrar grupos locales, globales y cuentas de usuario, formatear discos de servidores, mantener perfiles locales, auditar eventos, cerrar el servidor y administrar los recursos compartidos.

- Administradores del dominio. Es un grupo global cuyos miembros se añaden automáticamente al grupo local de administradores. Por tanto, todos los administradores de dominio son también miembros de Administradores, pero podemos quitarlos si no deseamos que tengan esos privilegios.

- Duplicadores. Es un grupo local que se crea tanto en servidores como estaciones de Trabajo Windows NT y permite administrar las duplicaciones de archivo en el dominio. Tiene un único miembro que debe ser una cuenta de usuario usada para iniciar sesiones en el servicio duplicador del controlador de dominio y de los controladores de reserva. No se deben agregar a este grupo cuentas de usuario reales.

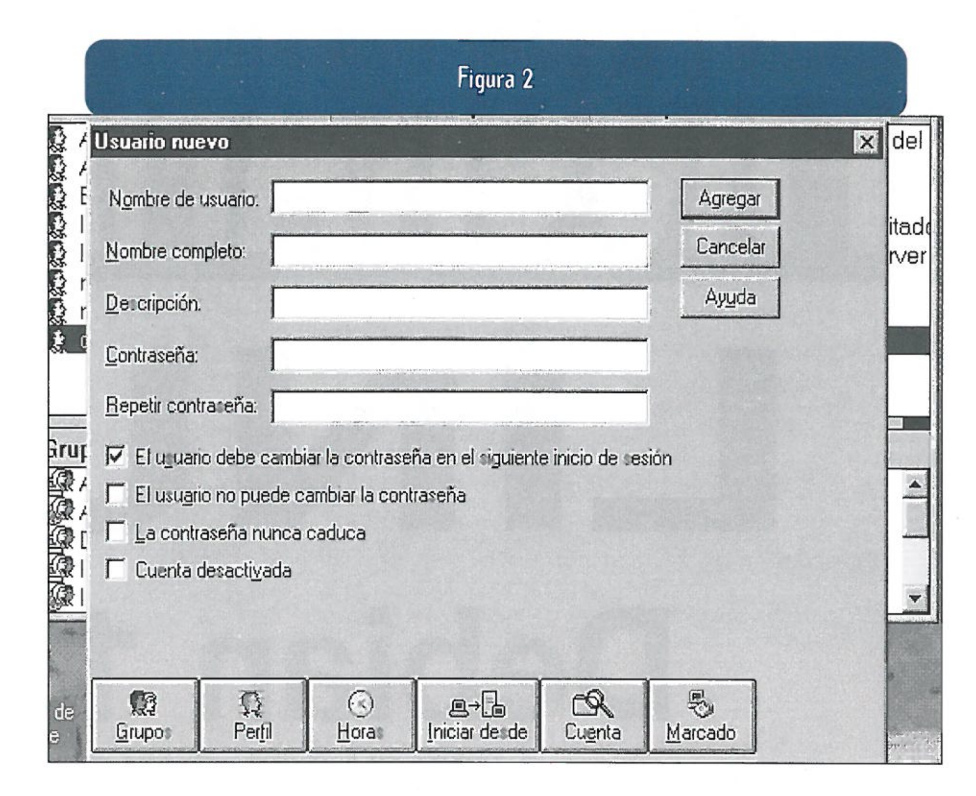


Figura 2

Existe un grupo global de invitados del dominio que pertenece por defecto al grupo local de invitados

- Invitados. Se trata de un grupo local que también se crea en estaciones de trabajo Windows NT. Permite que los usuarios ocasionales accedan al equipo o al dominio, pero con privilegios muy limitados. Existe un grupo global de invitados del dominio que pertenece por defecto al grupo local de invitados, pero puede quitarse.

- Operadores de copia. Pueden realizar copias de seguridad en el controlador principal de dominio y en los controladores de reserva. Para ello tienen permiso para iniciar sesión en los servidores y apagarlos.

- Operadores de cuenta. Grupo local que tiene permisos para utilizar el Administrador de usuarios para dominios. Pueden crear borrar y modificar usuarios, grupos globales y locales, así como añadir estaciones al dominio. No pueden asignar privilegios, administrar las cuentas de los administradores ni modificar los grupos de administradores, operadores de servidores, operadores de cuentas, operadores de impresión y operadores de copia.

- Operadores de impresión. Grupo local cuyos miembros pueden crear, eliminar y administrar los recursos compartidos de impresoras en los controladores principal y de reserva del dominio. También pueden iniciar sesiones en estos servidores y apagarlos.

- Operadores de servidores. Pueden administrar los controladores principal y de reserva de dominio, pero no pueden administrar su seguridad, es decir, pueden compartir y dejar de compartir los recursos, administrar las impresoras del dominio, hacer

Tabla 1. Derechos.

Derechos. y capacidades	Administradores	Operadores de servidor	Operadores de cuenta	Operadores de impresión	Operadores de copia	Todos	Usuarios Invitados	Invitados
Iniciar sesión localmente	Sí	Sí	Sí	Sí	Sí	No	No	No
Tener acceso a este equipo desde la red	Sí	No	No	No	No	Sí	No	No
Tomar posesión de archivos	Sí	No	No	No	No	No	No	No
Administrar el registro de seguridad y la auditoría	Sí	No	No	No	No	No	No	No
Cambiar la hora del sistema	Sí	Sí	No	No	No	No	No	No
Cerrar el sistema	Sí	Sí	Sí	Sí	Sí	No	No	No
Forzar el cierre desde un sistema remoto	Sí	Sí	No	No	No	No	No	No
Hacer copia de seguridad de archivos y directorios	Sí	Sí	No	No	Sí	No	No	No
Restaurar archivos y directorios	Sí	Sí	No	No	Sí	No	No	No
Cargar y descargar controladores de dispositivos	Sí	No	No	No	No	No	No	No
Agregar estaciones de trabajo a un dominio	Sí	No	Sí (con restricciones)	No	No	No	No	No
Crear y administrar cuentas de usuario	Sí	No	Sí (con restricciones)	No	No	No	No	No
Crear y administrar grupos globales	Sí	No	Sí (con restricciones)	No	No	No	No (si tienen permiso para acceder localmente al servidor)	No
Crear y administrar grupos locales	Sí	No	Sí	No	No	No	Sí	No
Asignar derechos de usuario	Sí	No	No	No	No	No	No	No
Administrar la auditoría de cuentas del sistema	Sí	No	No	No	No	No	No	No
Bloquear el servidor	Sí	Sí	No	No	No	Sí (si tienen permiso para acceder localmente al servidor)	No	No
Anular el bloqueo del servidor	Sí	Sí	No	No	No	No	No	No
Formatear el disco duro del servidor	Sí	No	No	No	No	No	No	No
Crear grupos comunes	Sí	Sí	No	No	No	No	No	No
Compartir y dejar de compartir directorios	Sí	Sí	No	No	No	No	No	No
Compartir y dejar de compartir impresoras	Sí	Sí	No	Sí	No	No	No	No

copia de seguridad y bloquear y apagar los servidores.

- Usuarios. No pueden acceder localmente a servidores NT ni realizar ningún tipo de tarea relacionada con la administración del sistema, pero poseen privilegios suficientes para realizar la mayor parte de las tareas que les sean necesarias.

Existe además un grupo global de usuarios del dominio. Este grupo global por defecto pertenece al grupo local de usuarios, pero es posible quitarlo.

Para obtener una aproximación más clara y detallada de lo que puede y no puede hacer cada grupo definido es posible consultar la tabla 1 de derechos.

Como podréis observar al consultar la tabla 1 de derechos, aparece un grupo adicional llamado Todos. Este grupo en realidad es una entidad especial, pero lo veremos más adelante en el artículo.

Como su nombre indica, todos los usuarios del sistema pertenecen a este grupo.

Administración de Usuarios

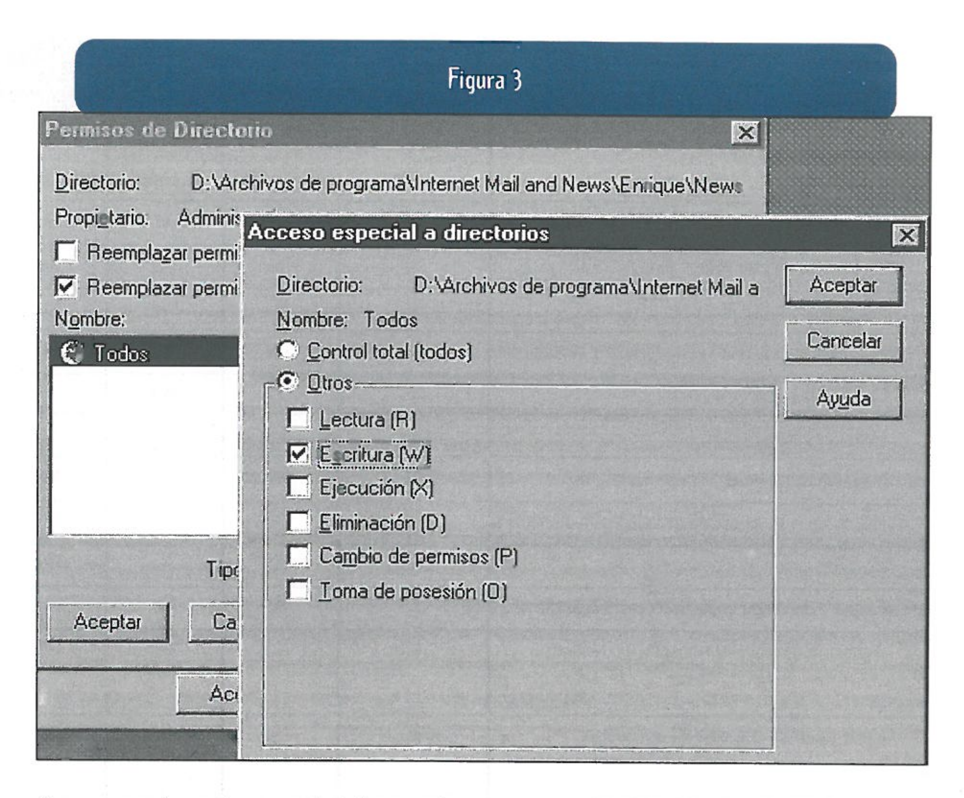
Los usuarios que pueden administrar a otros usuarios y grupos son los miembros de Administradores, Administradores de dominios y Operadores de cuenta (aunque éstos no pueden administrar las cuentas de administradores y operadores).

Para realizar la administración de usuarios se utilizará la utilidad Administrador de usuarios para dominios. A este respecto, hay un truco que nos permitirá administrar muy fácilmente las cuentas de varios dominios:

Para realizar la administración de usuarios se utilizará la utilidad Administrador de usuarios para dominios

La idea es crear tantos accesos directos de esta utilidad como dominios queramos administrar, aprovechando de paso la multitarea de Windows NT. Para ello abriremos el explorador de Windows NT y la carpeta \Winnt\Profiles\All Users\Menú Inicio\Programas\Herramientas administrativas (Común) y seleccionamos el Administrador de usuarios para dominios. Con este ítem seleccionado, pulsamos control+C y control+V para duplicar la entrada. Seleccionamos la copia recién creada, pulsamos F2 para cambiar el nombre y le ponemos el nombre que deseemos (normalmente el mismo con la coletilla del dominio al que apuntará). Con la nueva entrada seleccionada pulsamos el botón derecho del ratón y seleccionamos Propiedades. Pinchamos la pestaña Acceso directo, y en el apartado Destino añadimos al final el nombre del dominio al que apuntará (figura 1). Aceptamos y terminamos.

Figura 3



De este modo podremos administrar a la vez tantos dominios como queramos (o tantos como instancias del programa soporte el servidor).

Crear una cuenta de Usuario

Para crear una cuenta de usuario utilizaremos el Administrador de usuarios para dominios. Pincharemos en la opción Usuario nuevo del menú usuario y nos aparecerá la ventana de la figura 2. Podemos rellenar los siguientes campos:

- Nombre. Especificaremos un nombre único de login para la red, con un máximo de veinte caracteres. Es importante crear unas reglas para la formación de nombres de usuario. Si, por ejemplo, ponemos la letra inicial del nombre y el primer apellido deberíamos mantenerlo para el resto de los usuarios, ya que luego nos será de utilidad para localizarlo fácilmente. Si no mantenemos unas reglas será difícil recordar si a un usuario le hemos llamado EDiaz, EnriqueD, DiazE...

Es muy importante mantener unas reglas para la formación de nombres de usuario

- Nombre completo. Es un campo opcional para especificar el nombre real del usuario. Aquí también es conveniente tener una norma, esto es, empezar todos por el apellido o bien por el nombre.

- Descripción. Otro campo opcional utilizado para poner cualquier texto relativo a la cuenta de usuario.

- Contraseña. Palabra clave que se utilizará para iniciar la sesión. Puede tener hasta catorce caracteres, y se distinguen mayúsculas y minúsculas.

- Repetir contraseña. Requerido para verificar que la contraseña especificada en el campo anterior es correcta. De este modo conseguimos evitar que se pierda la contraseña debido a un error de producido al teclear.

- El usuario debe cambiar la contraseña en el siguiente inicio de sesión. Se especificará para obligar al usuario a cambiar la contraseña que le dio el administrador al darle de alta en la red.

- El usuario no puede cambiar de contraseña. Con esta opción evitamos que el usuario pueda modificar su contraseña.

- La contraseña nunca caduca. Esta opción prevalece sobre las especificaciones de duración de la contraseña y sobre el apartado anterior. El usuario debe cambiar la contraseña en el siguiente inicio de sesión. De este modo el sistema no obligará al usuario a cambiar la contraseña cada cierto tiempo.

- Cuenta desactivada. Se utiliza para impedir que alguien pueda utilizar los recursos de una cuenta. Cuando una cuenta está desactivada no se podrá iniciar sesión con ese nombre de usuario. Se utiliza, por ejemplo, para trabajadores discontinuos o que están de vacaciones.

Junto a estos campos encontraremos unos botones para especificar otras opciones de la cuenta:

- Grupos. Es utilizado para indicar los grupos a los que pertenecerá el usuario. Por omisión éste pertenecerá al grupo denominado Usuarios del dominio.

- Perfil. Lo utilizaremos para asignar un perfil de usuario, un archivo de comandos para el inicio de sesión o un subdirectorio particular para la cuenta del usuario.

- Horas. Nos permite especificar las horas a las que podrá conectarse el usuario. Podemos marcar con el ratón sobre una tabla los periodos de tiempo en los que permitiremos o denegaremos el acceso al usuario. La tabla es una matriz de días de la semana y horas del día, por lo que podremos establecer restricciones dependiendo del día de la semana. Por defecto no existen restricciones de conexión.

El usuario puede iniciar por defector una sesión desde cualquier estación de la red

- Iniciar desde. Indica desde qué estaciones podrá acceder el usuario a la red. Por defecto podrá iniciar sesión desde cualquier estación de la red, pero podemos establecer una lista de hasta 8 estaciones. Si establecemos esta restricción y el usuario inicia la sesión desde cualquier estación que no esté en la lista, su acceso a la red será rechazado.

- Cuenta. En este apartado podemos especificar una fecha de caducidad para la cuenta. Al llegar a dicha fecha, la cuenta se desactivará. También podemos especificar si la cuenta será global, esto es, para usuarios normales del dominio, o local si se trata de una cuenta creada en un dominio en el que no se confía y se desea dar ciertos privilegios a esa cuenta. Por defecto, la cuenta es global y no caduca.

- Marcado. Mediante esta opción permitimos que el usuario se conecte al servidor telefónicamente. Si le concedemos este privilegio, podremos establecer si el servidor le devolverá o no la llamada. Con la opción nunca el servidor no devolverá la llamada. Mediante la opción establecer por quién marca se producirá el siguiente proceso: el usuario se conecta al servidor, que comprueba su identidad, solicita un número de teléfono y cierra la conexión. Posteriormente el servidor llamará a ese número de teléfono y establecerá de nuevo la conexión, comprobando nuevamente la identidad.

Con la opción Preestablecer a, indicaremos un número de teléfono y se producirá el mismo proceso descrito anteriormente, a diferencia de que no solicitará un numero de teléfono. El usuario se conecta, el servidor valida su identidad y corta la conexión para llamar al teléfono que hayamos especificado, estableciendo nuevamente la conexión.

Si los permisos estándares no nos satisfacen podemos crear nuestro propio conjunto de permisos

Si vamos a crear varios usuarios que van a tener las mismas características, una buena opción es rellenar sólo los datos estrictamente personales (Nombre, Nombre completo y Contraseña) para establecer después conjuntamente sus características comunes. Para hacer esto los marcaremos en la pantalla principal del Administrador de usuarios para dominios y elegiremos la opción Propiedades del menú Usuario. Nos aparecerá prácticamente la misma pantalla que en la creación de usuarios, con la diferencia de que tendremos un cuadro con los nombres de los usuarios que estamos modificando.

Un buen detalle es que ahora los textos de las opciones nos aparecerán en plural.

Si estamos realizando modificaciones, mediante la opción Seleccionar Usuarios del menú Usuarios seleccionaremos grupos de usuarios — entendido como grupos existentes, como por ejemplo Invitados — y modificar sus cuentas.

Grupos, derechos, Permisos...

La creación de grupos no tiene ningún misterio. Simplemente hemos de darle un nombre y agregarle usuarios o grupos. Esto podemos hacerlo mediante las opciones Grupo local nuevo y Grupo global nuevo del menú Usuarios del Administrador de usuarios para dominios, o bien utilizando el asistente de administrador de grupos incluido en los asistentes administrativos.

Los volúmenes FAT sólo se pueden proteger a través de la red

Lo que sí tiene misterio es saber utilizar esos grupos. Por ello, vamos a recopilar aquí unas sencillas normas y recomendaciones que hemos ido dando a lo largo del artículo:

- Un grupo global sólo puede contener cuentas de usuario pertenecientes a un solo dominio. El término "global" indica que puede recibir permisos de múltiples dominios.
- Un grupo local puede contener cuentas de usuario y de grupo procedentes de múltiples dominios. El término "local" indica que sólo puede recibir permisos del dominio en el que se creó.
- Lo más eficiente es crear grupos globales para agrupar usuarios y otorgarles derechos mediante la inclusión de los grupos globales en otros locales. De este modo agregamos cuentas de usuario de los dominios de cuentas en los que se confía a los dominios de recursos que confían.
- Es preferible dar permisos a grupos antes que a usuarios. De este modo si creamos otro usuario con iguales necesidades sólo necesitaremos añadirlo al grupo. Procediendo de esta manera no sólo nos facilitamos la creación sino también posteriores modificaciones.

También hay que tener muy claro que los permisos y los derechos no son la misma cosa: los derechos se aplican a todo el sistema mientras que los permisos se aplican sobre objetos concretos, como archivos, directorios o impresoras.

Los derechos no están asociados a objetos, por lo que en caso de conflicto prevalece el derecho. Por ejemplo, el derecho del operador de copia prevalece sobre las protecciones contra lectura de archivos y directorios. Una vez aclarado esto, que es muy importante para saber exactamente qué puede y qué no puede hacer un usuario, veamos cuáles son los derechos y cómo determinan las tareas que se pueden realizar en la red.

- Acceder a este equipo desde la red. Permite a un usuario conectarse al servidor a través de la red.

El derecho del operador de copia prevalece sobre las protecciones contra la lectura de archivos y directorios

- Administrar los registros de auditoría y seguridad. Permite establecer los tipos de accesos a los recursos del sistema que se van a auditar. Sin embargo, este derecho no permite que un usuario establezca la auditoría del sistema usando el comando Auditoría del menú Directivas del Administrador de usuarios para dominios. Esta capacidad sólo la mantiene el grupo Administradores.
- Agregar estaciones de trabajo a un dominio. Permite agregar una estación al dominio, de manera que la estación reconozca las cuentas del propio dominio y de aquellos en los que se confía.
- Apagar el sistema. Permite apagar el servidor.
- Cambiar la hora del sistema. Establecer la hora del reloj interno del equipo.
- Cargar y descargar controladores de dispositivo. Instalar y quitar controladores de dispositivo.
- Hacer copias de seguridad de archivos y directorios. Hacer copias de seguridad del sistema. Permite al usuario leer todos los archivos, y prevalece sobre los permisos de archivo y directorio.
- Iniciar sesión como servicio.
- Inicio de sesión local. Permite al usuario iniciar sesión desde el propio servidor, esto es, desde su teclado.
- Restaurar archivos y directorios. Permite restaurar las copias de seguridad. Permite al usuario escribir todos los archivos, y al igual que en el caso de las copias, prevalece sobre los permisos de archivo y directorio.
- Tomar posesión de archivos y otros objetos. Permite tomar la posesión de archivos, directorios y objetos, obteniendo así permisos especiales sobre ellos.

Además de éstos permisos existen otros derechos de usuario avanzados. Son útiles para administradores y programadores, y normalmente no los asignaremos ya que con algunos de estos derechos hay que saber muy bien lo que se hace. Estos derechos son:

- Actuar como parte del sistema operativo. El sistema nos tomará entonces como parte segura del sistema operativo.
- Bloquear páginas en memoria. Permite bloquear páginas en memoria de manera que no puedan ser paginadas a un archivo del disco.
- Crear archivo de paginación. Crear otro archivo de paginación.
- Crear objetos compartidos permanentes. Crear objetos especiales como \Device, que se utilizan dentro de la plataforma de Windows NT.

Para establecer los derechos de usuarios utilizaremos el Administrador de Usuarios para Dominios

- Crear un objeto testigo. El usuario o un programa ejecutado por él pueden crear testigos de acceso hacia un objeto.
- Depurar programas. Permite que un usuario depure objetos de bajo nivel, como subprocesos.
- Generar auditorías de seguridad. Permite que un usuario o programa genere entradas para el registro de auditoría de seguridad.
- Incrementar prioridades de planificación de sucesos. Permite al usuario incrementar la prioridad de un suceso.
- Iniciar sesión como proceso por lotes. El usuario iniciará la sesión utilizando la cola para procesos por lotes para inicios de sesión retardados.
- Iniciar sesión como servicio. Permite que un proceso se registre en el sistema como servicio. Se usa para administrar el servicio duplicador de directorios.
- Modificar valores de entorno de la memoria no volátil. Permite variar las variables de entorno del sistema (mucho cuidado con esto, ya que suele confundirse con modificar las variables de entorno de usuario).
- Perfilar el rendimiento del sistema. Permite usar las capacidades de registro de actividad para el sistema.
- Perfilar un proceso individual. Igual que la anterior, pero el objeto será un proceso y no el sistema.
- Reemplazar un testigo a nivel de proceso. Permite modificar un testigo de acceso de seguridad de un proceso.
- Saltarse la comprobación de recorrido. Permite cambiar hacia cualquier directorio, incluso si no se tienen permisos para acceder a los directorios.

Para establecer los derechos de usuario utilizaremos el Administrador de usuarios para dominios, eligiendo la opción Derechos de usuario que se encuentra bajo el menú Directivas. Por defecto los derechos avanzados no aparecen, pero podemos verlos marcando la casilla de verificación Mostrar derechos de usuario avanzado.

Tabla 2. Permisos.

Permiso	Sin acceso	Lectura	Cambio	Control Total
Mostrar nombres de subdirectorios y archivos	No	Sí	Sí	Sí
Mostrar datos y atributos de archivos	No	Sí	Sí	Sí
Ejecutar archivos de aplicación	No	Sí	Sí	Sí
Ir a subdirectorios del directorio	No	Sí	Sí	Sí
Crear subdirectorios y agregar archivos	No	No	Sí	Sí
Cambiar datos y añadir datos a archivos	No	No	Sí	Sí
Cambiar atributos de archivos	No	No	Sí	Sí
Eliminar subdirectorios y archivos	No	No	Sí	Sí
Cambiar permisos (NTFS)	No	No	No	Sí
Tomar posesión (NTFS)	No	No	No	Sí

Permisos

La seguridad del acceso a los volúmenes se administra a través de los permisos. En los volúmenes NTFS podemos establecer permisos en el ámbito de directorio y en el ámbito de archivo, y en ellos estableceremos qué usuarios y grupos tienen acceso y qué nivel de acceso les

La seguridad de acceso a los volúmenes se administra a través de los permisos

permitiremos. También podremos establecer otro nivel en los permisos de los recursos compartidos, estos permisos funcionarán conjuntamente a los permisos de archivo y directorio, y se aplican a las conexiones a través de la red.

Estos permisos son los que permiten a un usuario de la red conectar con el recurso compartido y definir su nivel de acceso. Combinando estos permisos obtenemos una forma fácil de administrar la seguridad: permitiremos el acceso a Todos (la entidad Todos) y luego aplicaremos los permisos de directorios y archivos. Aclaremos un poco estos conceptos:

Podemos bloquear los permisos de recurso compartido con los permisos de archivo y directorio

Cuando queremos compartir un directorio se hace con un nombre de recurso compartido. Este recurso compartido puede tener el mismo nombre que el directorio (lo más usual), pero no tiene por qué ser así. Los permisos de recurso compartido se aplican sobre el recurso que compartimos y podemos bloquearlos con los permisos de archivo y directorio. Por ejemplo, podemos dar control total al recurso compartido APPS y bloquear el acceso a un hipotético subdirectorio PROC aplicando permisos de directorio.

Con los volúmenes FAT el asunto cambia sustancialmente. Podremos proteger los archivos bajo FAT únicamente a través de la red y sólo si el directorio está compartido, es decir, sólo podemos aplicar los permisos de recurso compartido. Una vez compartido un directorio, podremos protegerlo especificando un conjunto de permisos para recursos compartidos que sólo afectará a los usuarios que conectan con el directorio compartido a través de la red.

Recursos y Entidades

Durante la instalación de Windows NT se crean una serie de recursos compartidos que no es conveniente modificar, ya que se crearon para uso administrativo y del sistema:

Letra_unidad$. Se utiliza para la administración remota del sistema y representa una unidad del servidor. En lugar de conectar unidades de red y tener que recordar que Z: se corresponde con C:, simplemente tecleamos C$.

Admin$. Es un recurso utilizado por el sistema durante la administración remota, y se corresponde con el directorio Winnt, o aquel donde hayamos realizado la instalación del sistema operativo.

PC$. Comparte las canalizaciones con nombre que son esenciales para la comunicación entre programas. Se utiliza durante la administración remota de un equipo y cuando se ven los recursos compartidos de un equipo.

NETLOGON. Se utiliza durante el inicio de sesión de red en los controladores de dominio para procesar las peticiones de inicio de sesión de dominio.

PRINT$. Permite el acceso a las impresoras compartidas.

REPL$. Recurso creado por el sistema cuando un equipo con Windows NT se configura como servidor exportador de duplicación. Es necesario para exportar la duplicación.

Asimismo, durante la instalación también se crearon una serie de entidades especiales a las que podemos otorgar permisos. Estas entidades especiales son las siguientes:

Todos representa todos los usuarios actuales y futuros de la red.

Sistema representa el sistema operativo local. Tiene permisos sobre algunos directorios y archivos, y éstos permisos nunca se deben revocar. No es necesario darle permisos sobre otros directorios, a menos que los utilice el sistema operativo.

Red representa a todos los usuarios actuales y futuros que tienen acceso a directorios a través de la red, mientras que Interactivo representa a todos los usuarios que acceden al directorio en el propio servidor. Si especificamos permisos distintos para Red e Interactivo, un mismo usuario tendrá permisos diferentes dependiendo de que el acceso sea local o a través de la red.

Es posible establecer las características comunes de varios usuarios en una sola operación

Creador propietario representa a los usuarios que crearán directorios o archivos en el directorio activo, es decir, si alguien crea un directorio o archivo, automáticamente tendrá para ese archivo los permisos que hayamos establecido para Creador propietario.

Funcionamiento de los Permisos

Antes de compartir un recurso, deberíamos establecer permisos individuales para el directorio y sus archivos. Para facilitarnos la tarea existe un conjunto de permisos estándar. Éstos permisos no son otra cosa que las combinaciones más usuales de permisos individuales. Los permisos individuales son: lectura (R),

escritura (W), ejecución (X), borrado (D), cambio de permisos (P) y toma de posesión (O).

Los permisos estándar son:

- Sin acceso. No se tiene acceso al directorio, ni siquiera si el usuario pertenece a un grupo que sí tiene acceso al directorio.
- Listado (RX). Se podrá listar el contenido del directorio y cambiar a los subdirectorios existentes, pero no tendrá acceso a los nuevos archivos creados en él.
- Lectura (RX). Permite leer el contenido de los archivos en el directorio y ejecutar sus aplicaciones, aplicado en el ámbito de archivo, permite leer el contenido del archivo y ejecutarlo si es una aplicación.
- Adición (WX). Se permite añadir archivos al directorio, pero no ver el contenido del directorio.
- Adición y lectura (RWX). Se permite la adición y lectura de los archivos del directorio, pero no modificarlos.
- Cambio (RWXD). Se permite leer, agregar, modificar y borrar archivos.

Si los permisos estándar no nos sirven podemos crear nuestros propios permisos

- Control total. Se permite leer, añadir, modificar, borrar, cambiar los permisos y tomar posesión de archivos y subdirectorios.

Si ninguno de los permisos estándar nos viene bien, podemos crear nuestro propio conjunto de permisos. Para ello haremos clic con el botón derecho del ratón en una carpeta, elegimos la opción propiedades y seleccionamos la pestaña seguridad. Pulsamos sobre el botón permisos, desplegamos el cuadro de lista de Tipo de acceso y seleccionamos Acceso especial a directorios o Acceso especial a archivos. Se nos abrirá una ventana en la que podemos especificar nuestro propio conjunto de permisos (figura 3). En el caso de la figura hemos elegido escritura para que el usuario pueda crear nuevos archivos, pero no pueda hacer nada más que eso.

Cuando establezcamos los permisos debemos tener en cuenta una serie de normas que nos serán de utilidad a la hora de administrar la seguridad:

- Los permisos son acumulables. Si un usuario es miembro de un grupo que tiene permiso de listado y de otro grupo que tiene permiso de cambio, el usuario tendrá permiso de cambio (es superior a listado). Esto no ocurre así con el permiso Sin acceso, que prevalece sobre todos los demás. Si en el caso anterior uno de los grupos tuviera especificado Sin acceso, el usuario no podría acceder al directorio.
- Los permisos se heredan. Cuando agregamos un archivo en un directorio, heredaremos sobre el archivo los mismos permisos que teníamos para el directorio.
- El creador es propietario. Cuando creamos un nuevo archivo o directorio, nos convertimos en sus propietarios. Esto se puede traducir en que podemos modificar los permisos establecidos sobre el archivo. Los administradores pueden tomar posesión de cualquier archivo o directorio.
- Los permisos de archivo prevalecen sobre los de directorio. Si tenemos permiso de lectura sobre un directorio, pero no sobre un archivo contenido en el directorio, no podremos leer el archivo.
- Lo mejor es conceder los permisos sobre grupos y no sobre usuarios individuales. De este modo si deseamos quitarle permisos bastará con quitarle del grupo. Resulta mucho mejor y más práctico que ir explorando todos los directorios archivos sobre los que tenía permisos y quitarlos directorio por directorio y archivo por archivo.

En cuanto al establecimiento de permisos sobre recursos compartidos se hace de igual forma que para los permisos de directorios y archivos, pero los permisos que especifiquemos serán válidos sólo cuando el usuario utilice el recurso a través de la red. Podremos aplicar cualquiera de los cuatro tipos de permisos para recursos compartidos: Control total, Cambio, Lectura o Sin acceso. Si trabajamos con volúmenes NTFS lo más sencillo es establecer los permisos para los directorios y archivos y luego asignar Control total para Todos a la hora de compartirlo.

Únicamente podremos proteger los volúmenes FAT como recurso compartido

La seguridad del sistema no se verá comprometida, ya que se aplicarán los permisos de directorio. Esto no ocurre así con volúmenes FAT, ya que sólo podremos proteger estos volúmenes como recurso compartido.

Para terminar, los permisos para recursos compartidos son válidos además de los permisos NTFS establecidos en el directorio, pero si entran en conflicto se aplican los más restrictivos. Si especificamos Cambio en el ámbito de directorio y Control total en el ámbito de recurso compartido, se aplican los de cambio; y si especificáramos Cambio en el ámbito de directorio y Lectura en el ámbito de recurso compartido se aplicarán los de Lectura.

ASN 1. Abstract Syntax Notation 1

Debido a la infinita variedad de máquinas, sistemas operativos, protocolos, etc., existentes en la red surgió la necesidad de definir un lenguaje de definición de objetos, mediante el cual las aplicaciones pudieran entender los datos, por utilizar todas el mismo lenguaje, a la hora de definirlos.

ASN.1 pertenece al modelo OSI (*Open Systems Interconnection*) definido por la ISO (*International Standards Organization*) y es utilizado por SNMP (*Simple Network Management Protocol*). Internet, como todos sabemos, tuvo sus orígenes en la red experimental Arpanet. Al aumentar el número de máquinas conectadas, hubo que hacer algo para poder gestionarla, así, en 1990 se publicó la RFC1157 donde se definía el SNMP, mediante el cual se pretendía dar solución a los problemas de gestión y control de red. El SNMP adoptaba como lenguaje de definición de datos el ASN.1.

Dentro del modelo OSI es en el nivel de aplicación donde se empleará dicho lenguaje a la hora de definir las estructuras de datos, con las que se comunican las aplicaciones de dicho nivel. Una completa definición de ASN.1 se encuentra en el estándar ISO 8824.

ASN.1

Situados pues, empecemos a ver en qué consiste y las particularidades de dicho lenguaje. En primer lugar hay que decir que ASN.1 es un lenguaje que distingue entre letras mayúsculas y minúsculas y que uno de los tipos de datos, los primitivos, tienen todas las letras en mayúsculas, mientras que, por el contrario, los tipos que definimos nosotros sólo tienen que tener la primera letra del identificador en mayúsculas. La manera en la que podemos asignar un nombre a un tipo es:

```
Nombre-de-tipo ::= tipo.
```

Para familiarizarnos con la sintaxis de ASN.1, comencemos por estudiar los tipos primitivos del lenguaje.

Tipos primitivos

Este conjunto de tipos son los que nos encontramos habitualmente en cualquier lenguaje de programación: enteros, booleanos, reales, enumerados, cadenas de caracteres, etc. Los iremos viendo indicando las particularidades de cada uno de ellos. Es importante mencionar que con ASN.1 se logra que, por ejemplo, dos tipos enteros, cada uno en una máquina diferente, distinto número de bits, distinto tipo de codificación, etc. resulten compatibles para las aplicaciones.

El conjunto de tipos primitivos definidos en ASN.1 es el siguiente: BOOLEAN, INTEGER, REAL, BIT STRING, OBJECT IDENTIFIER, OCTECT STRING y ENUMERATED. Como dijimos anteriormente, todos deben ir en mayúsculas. Estudiemos cada uno de los tipos.

• Tipo BOOLEAN. Sólo puede tomar dos valores TRUE o FALSE. Veamos con un

¿TE ATREVES?

Desde **Sólo Programadores** estamos preparándonos para un torneo muy especial, de ordenador a ordenador, de programador a programador. El torneo de ajedrez está empezando a tomar forma y éstas son las bases, aunque también podéis informaros en AWS en la calle Serrano Jover, 3, de Madrid, o en el teléfono (91) 5 42 50 87. Para conocer cuál será el alcance de la participación, os pedimos que os inscribáis enviando vuestros datos a la dirección de AWS o a la dirección de e-mail: **solop@towercom.es.** El periodo de inscripción ya ha comenzado y terminará el 30 de noviembre, y la fecha límite para la recepción de los programas es el 15 de diciembre de 1997. La celebración del torneo será anunciada con suficiente antelación y podréis participar en persona o desde vuestra propia ciudad, ya que vuestro programa os representará y en todo momento estará presente un notario.

BASES DEL TORNEO

1. Se presentará un programa que juegue al ajedrez recibiendo la jugada del contrario y devolviendo la propia. El tiempo de la partida será medido por un juez y la partida se jugará físicamente en una mesa con un tablero siguiendo las instrucciones que cada ordenador dé; cada ordenador avisará de que ha elegido un movimiento con un pitido y mostrará su jugada en la pantalla de la forma: casilla a mover - casilla destino. La notación de la casilla vendrá dada por su columna y su fila. Para las columnas se utilizarán las de la A a la H siendo la columna A la primera columna de la izquierda, visto el tablero desde la posición de las blancas. Para las filas utilizaremos los números del 1 al 8 siendo la fila 1 la más cercana a nosotros (visto el tablero desde el lado de las blancas). Como ejemplo, el movimiento del peón de rey dos lugares hacia adelante sería: "E2 - E4". La jugada que hace el contrario se anotará en el ordenador de forma manual tecleando "E2-E4" <RETURN>.
2. El programa jugará al ajedrez según las normas de este juego, admitiendo todos sus movimientos.
3. El programa se ejecutará bajo Windows 95, MS Dos 6.22 y Linux, en un Pentium 166 con 16 MB de RAM.
4. El programa podrá jugar con las blancas o con las negras indistintamente.
5. Se debe entregar el código fuente, la versión del compilador utilizado y una breve documentación para su utilización.
6. Un programa podrá ser descalificado si realiza un movimiento no válido y esto deberá ser advertido por el contrario; por lo tanto, un movimiento será válido siempre que el otro programa no nos advierta de lo contrario. De la misma forma, si un programa nos advierte de un movimiento erróneo y éste no lo es, será descalificado.
7. Los jueces son los encargados de advertir las posiciones de tablas, el programa no advertirá de ello.
8. El programa debe detectar si ha ganado la partida y quedará descalificado ante cualquier detección errónea. No se avisará al contrario en el caso de que deje al rey al descubierto y ganará la partida el que antes mate al rey.
9. La metodología del concurso se reservará hasta que sea conocido el número de participantes (eliminatoria, puntos, etc.)
10. No se podrán utilizar en el programa bases de datos adicionales.
11. Los derechos del programa ganador pasarán a la propiedad de la empresa Tower Communications, SRL.
12. El premio consistirá en un Pentium PRO 200 con 64 MB RAM, VIRGE4 MB, CDX 16, disco duro 5 GB, monitor 17", teclado, ratón y disquetera.

ANEXO A LAS BASES

1. La velocidad de juego se establece en 2 horas (1 hora por jugador).
2. Las jugadas que el árbitro considere erróneas eliminarán al programa que las haya indicado.
3. El programa debe ser original.
4. La fecha de recepción del fuente y el compilador se establece entre el 1 y el 31 de Enero. Se deberán remitir a la siguiente dirección: AWS Informática, Serrano Jover 3, 28.015, Madrid, a la atención de Fernando Núñez.
5. El primer día de celebración del torneo será el 8 de Febrero de 1998 en la dirección indicada en el punto anterior.
6. Debido a las peticiones recibidas por los participantes, se admiten bases de datos adicionales.

CONCURSO DE PROGRAMACIÓN

SÓLO PROGRAMADORES Y AWS INFORMATICA

SERRANO JOVER 3, MADRID

ejemplo cómo podemos definir un tipo buleano y asignar un valor a una variable buleana:

```
Acierto ::= BOOLEAN
— Acierto ahora es un nuevo tipo, e
— igual al buleano.

respuesta Acierto ::= TRUE
— respuesta es de tipo Acierto y vale true.
```

Fijémonos en esos dos guiones "—", indican que lo que viene a continuación es un comentario.

• Tipo INTEGER. Ya conocido por todos, define los números enteros, desde menos infinito a más infinito, en teoría sin límite, aunque con límite en la práctica Igual que en el caso anterior veamos unos ejemplos.

```
Libros ::= INTEGER {pocos (100) ,
                    muchos (10000)}
— Libros, tipo entero.
biblioteca Libros ::= pocos
— biblioteca es de tipo entero y vale 100.
```

Podemos observar que hemos creado unos alias para 100 (pocos) y para 10000 (muchos). De esa manera en ASN.1 se pueden asignar nombres a determinados valores concretos.

• Tipo BIT STRING. Como su nombre indica se definen así arrays de bits de cualquier longitud. En este tipo se encuentran maneras de asignar valores muy curiosas:

```
Cadenabit ::= BIT STRING
binarios Cadenabit ::= '1001001' B
— La B nos indica formato binario.
anuarios Cadenbit ::= 'ABC00F' H
— La H nos indica formato hexadecimal.
tresbits Cadenabit ::= { prime (0),
                    segun (1), terce (2) }
— El número entre paréntesis indica el peso la
— cadena de bits y el nombre que le damos.
```

• Tipo REAL. Este tipo de números viene representados por la mantisa, la base y el exponente de la siguiente forma:

```
num REAL ::= { mantisa, base, exponente }
— por ejemplo...
num REAL ::= { 27182818, 10, -7 }
— num es igual al numero e = 2'7182818
```

• Tipo ENUMERATED. Representa el típico tipo conjunto, integrado por un número finito de elementos.

```
Figuras ::= ENUMERATED { círculo (0), triángulo
        (1), cuadrado (2) }
— Tipo con tres valores posibles
forma Figuras ::= cuadrado
— forma toma el valor 2
```

• Tipo OCTECT STRING. Cadenas de octetos, cada octeto varía entre cero y doscientos cincuenta y cinco.Veamos varios ejemplos:

```
Cadena ::= OCTECT STRING
— Cadena, tipo Octect String
nombre Cadena ::= "Solo Programadores"
bits Cadena ::= '11100111 00001111' B
— dos octetos expresados en binario.
```

•Tipo OBJECT IDENTIFIER. Este tipo es propio de ASN.1 y se emplea para identificar a los objetos OSI. Su estructura consiste en cadenas de números, los cuales, según su posición en la cadena, indican los grupos y subgrupos en los que están estructurados los tipos OSI. En un primer nivel están los tipos asociados a ISO (0), los del ITU antes CCITT (1), en un segundo nivel se encuentra los asociados a estas organizaciones (2). A su vez estos tres grupos se componen de otros y éstos de otros más, constituyendo una estructura arborescente. Veámoslo mas claro con un ejemplo.

ASN.1 se emplea para la definición de estructuras de datos en el nivel de aplicación de la torre OSI

```
Obj ::= OBJECT IDENTIFIER
dato Obj ::= { 1361212 }
— El uno representa al ISO el tres a identified
— organization, el seis a dod, el uno a internet,
— eldos a mgmt, el uno a mib-2 y así
— sucesivamente.
```

Tipos Predefinidos

Los tipos predefinidos, sin ser primitivos, se pueden utilizar sin haber sido definidos previamente, el lenguaje los incorpora por su frecuente uso, ahorrando trabajo al programador. Los más representativos son: NumericString, IA5String, CharacterString, ObjectDescriptor, GraphicString, UTCTime, GeneralString, TeletexString, etc.

OBJECT IDENTIFIER es un tipo propio de ASN.1 que se emplea para identificar a los objetos OSI

De todos estos cabe señalar a IA5String, compuesto por cadenas de caracteres del alfabeto ASCII, a UTCTime, como el tipo que representa a la hora y fecha, en un formato ampliamente utilizado.

Tipos construidos

Por fin llegamos a los tipos que el programador puede definir para representar la información según sus necesidades. Los tipos construidos son cinco, SEQUENCE y SET, SEQUENCE OF y SET OF y por último tipo CHOICE.

Los dos primeros se usan para construir tipos constituidos por un conjunto de tipos y se diferencian entre sí en que en el primero es importante el orden de los campos, mientras en los set no, esto

quiere decir que a la hora de enviar los datos en uno es importante el orden en que se envían y en el otro no.

En los dos siguientes, *sequence of* y *set of*, ocurre lo mismo respecto al orden de envío, pero se diferencian del caso anterior en que los tipos que los integran son todos iguales. Por último, el tipo CHOICE consiste en una unión de las de C, es decir, una misma zona de memoria puede adoptar diferentes tipos, los pertenecientes a la definición del tipo *choice*. A la hora de enviar el dato se añade una etiqueta, que es el nombre de la variable, para interpretar correctamente el dato recibido.

Estas construcciones tienen la misma estructura: primero su nombre, después "::=", seguidamente el tipo de estructura *set, set of, choice*, etc. Por último, entre llaves, los distintos campos que la compongan, separados por comas. Cada campo será un nombre de variable seguido por su tipo. Este podrá ser cualquiera, primitivo, construido, predefinidos, etc.

Un ejemplo muy simple sería el siguiente:

```
Patrimonio ::= SEQUENCE
{
      metálico ::= INTEGER,
      coche::= IA5String,
      casa::= OCTECT STRING,
      banco ::= INTEGER
}
```

y a una variable tipo Patrimonio se le puede asignar un valor de la siguiente manera:

```
manolo Patrimonio ::=
{
      metálico 100000,
      coche "Porche",
      casa "Avenida locos de la informática num.
      5",
      banco 1000000
}
```

De los tipos nos queda comentar la existencia de subtipos, los cuales consisten en rangos de valores de otros tipos, algo así como Mes ::= INTEGER (1..31). Claramente observamos cómo los números enteros quedan ahora reducidos a 31.

Para terminar con la explicación sobre los tipos en ASN.1 nos queda mencionar dos posibilidades que nos permite el lenguaje: ANY y EXTERNAL. El primero consiste en la posibilidad de que la variable definida como ANY pueda adoptar cualquier tipo válido en el lenguaje. External se refiere al caso en el que no podamos definir nuestras estructuras con ASN.1, y lo consigamos hacer con otro lenguaje, o al caso de poder importar definiciones de otros módulos.

Los módulos son conjuntos de definiciones de tipos y estructuras a las que se les asocia un nombre y que forman una entidad propia. Finalmente dos palabras reservadas: OPTIONAL, que nos indica que un campo en una estructura puede o no estar presente a la hora de asignar valores a la variable, y DEFAULT, que se emplea en el momento de definir un tipo, para asignar valores por defecto a las variables. Hasta ahora hemos visto que primero definimos el tipo y posteriormente le asignamos valores.

Etiquetas

ASN.1 es un lenguaje orientado a la definición de estructuras de datos que tiene una particularidad: su función principal es conseguir que dos máquinas diferentes, tanto en hardware como en software, sean capaces de enviarse datos. Estos datos no son bits sueltos, o el texto de un informe en algún lenguaje natural, sino estructuras complejas de datos ya que, como hemos comentado antes, un tipo de datos tan simple como un entero puede tener una representación completamente diferente en una máquina y en otra.

En este proceso de compartición de datos se realiza el envío, y con él, aparece el problema de decidir cuándo se manda algo, y qué es lo que se manda. Para resolver esto ASN.1 cuenta con las etiquetas. Los tipos primitivos, predefinidos y los constructores, tienen cada uno una etiqueta estándar que los identifica. Dicha etiqueta no aparece en el momento de la definición de los tipos o valores y sólo se usa en el momento de la transmisión, a no ser que la pongamos nosotros mismos. Por lo tanto con las etiquetas podemos reconocer en recepción el tipo de datos que nos llegan.

Dos tipos Integer en máquinas distintas pueden tener aspectos completamente distintos

Una etiqueta consta de dos campos, el primero el tipo de etiqueta y el segundo un número entero, ambos entre corchetes. En ASN.1 tenemos cuatro tipos de etiquetas: [UNIVERSAL n], [APPLICATION n], [PRIVATE n], [n], esta última no contiene nombre, sólo un número y se las llama específicas de contexto. Una etiqueta se coloca antes del nombre del tipo que va a identificar.

Las etiquetas universales son válidas en cualquier aplicación, como la de los tipos primitivos. El tipo INTEGER tiene la misma etiqueta en cualquier aplicación. Las etiquetas *application* tienen un ámbito más restringido, que se circunscribe a una aplicación concreta, fuera de la cual no indica nada. Las privadas pueden ser válidas en varias aplicaciones, pero con la restricción de un ámbito concreto: una compañía, un país, etc. Por último, las específicas son las de ámbito más restringido, por ejemplo un tipo construido. A su vez a las etiquetas se les puede añadir otro aspecto, que sean IMPLICIT o EXPLICIT. Todas las etiquetas son de tipo *explicit*, a no ser que queramos que sean *implicit* para lo cual indicamos esta particularidad entre la etiqueta y el tipo que identifica. La diferencia entre ambos aspectos es que, en el momento de la transmisión, sólo se mande la etiqueta que hemos puesto y no la del tipo.

Recordemos que cada tipo tiene una etiqueta que lo identifica. Por lo tanto con *implicit* sólo se manda la etiqueta nuestra y si no ponemos nada se envían las dos, la nuestra y la del tipo. Escribamos unas cuantas etiquetas para centrar las ideas.

```
Compra ::= [ APPLICATION 1 ] IMPLICIT SET {
        dinero [ 0 ] IMPLICIT INTEGER,
        verdura [ 1 ]
}
```

Como hemos dicho antes, las etiquetas tienen que ver con la información que se envía a la hora de transmitir los datos a la aplicación destino. Por lo tanto su funcionamiento lo veremos en la parte complementaria de ASN.1: las BER definidas en la ISO 8825. Observamos entonces que en ASN.1 se cuelan aspectos de la transmisión de los datos. Éstos deberían ser aspectos separados, por un lado la definición y por otro las reglas de envío, sin embargo los dos estándares se solapan.

Sintaxis de Transferencia

Las reglas BER (*Basic Encoding Rules*) complementan al estándar ASN.1 a la hora de codificar los datos para poder ser enviados, de forma que el receptor interprete correctamente los datos que le llegan, sin ambigüedades. Cualquier valor enviado tiene los siguientes campos:

1.- Tipo de dato
2.- Longitud del campo de datos expresado en bytes
3.- Valor de dicho dato
4.- Si la longitud no es conocida, un flag indicando el fin de los datos

Campo tipo

El campo tipo suele tener un tamaño de un byte y está dividido en tres campos: Clase, P/C y Etiqueta. El primero consta de dos bits y puede tomar cuatro valores, los cuatro tipos de etiquetas, Universal 00=0, Application 01=1, Private 10=1 y por último Específica 11=2, por lo tanto el campo Clase indica el tipo de etiqueta del dato. El campo P/C consta sólo de un bit y se limita a indicar si el campo es construido (1) o primitivo (0); el tercer campo, etiqueta, lo componen los cinco bits restantes del byte y por lo tanto el mayor número que soporta es 31. En caso de etiquetas de mayor tamaño, las reglas BER nos dicen lo siguiente: añadir tantos bytes como

Las etiquetas tienen que ver con la información que se envía a la hora de transmitir los datos a la aplicación destino

sea necesario, todos empezando por 1 menos el último que empezaría por cero. Este primer bit de los bytes añadidos no computa en el valor de la etiqueta, el peso de los bytes decae de izquierda a derecha como los dígitos de un número decimal. Por último indicar que, si el tipo es construido, el campo valor contiene una estructura como si de un nuevo dato se tratara, y así de forma recursiva.

Campo longitud

Este campo indica la longitud, expresada en bytes, del campo datos, lo cual representa un inconveniente a la hora de codificar pues hasta que no tenemos el dato completamente codificado no sabemos su longitud en bytes. El campo longitud está compuesto por un byte donde el primero por la izquierda es cero y los siete restantes se utilizan para indicar la longitud. Es posible que el tamaño del campo valor exceda de 126, máximo valor que podemos indicar con siete bits, en ese caso lo resolveremos de la siguiente manera. En el primer bit del primer byte en vez de un cero ponemos un uno, lo cual indica que la longitud excede a 126 y en el resto de bits ponemos el número de octetos necesarios para indicar la longitud. Es en esos octetos donde ponemos la longitud. Cuando la longitud es indefinida basta poner en el campo longitud un uno seguido de siete ceros, el marcador de fin de datos consiste en dos bytes los dos a cero, lo que invalida la posibilidad de que en el campo datos tengamos dos bytes puestos a cero pues se confundiría con el marcador fin de datos.

Campo datos

Es aquí donde se pone el valor del dato a transmitir, si es primitivo se pone su valor según veremos. Cada tipo tiene su manera especial de codificación, pero si se trata de un tipo construido la cosa se complica, ya no tenemos un valor a codificar, tenemos varios y para cada uno tenemos que empezar como si se tratara de una nueva codificación. Es decir, para un tipo construido con dos campos, primero pondríamos el tipo de dato construido, su longitud no la sabremos hasta haber codificado todo el dato completo, y dentro de valor empezamos poniendo el tipo longitud y valor del primero, y luego del segundo. Si contamos los bytes sabremos la longitud del dato construido completo. Es posible que dentro de datos construidos tengamos a su vez datos construidos, procederíamos de igual forma, algo así como un proceso recursivo.

Veamos ahora cómo se codifican los distintos tipos primitivos.

• Tipo Integer. Uno de los más sencillos, por ejemplo el número entero 16 se codificaría como (002 01 00010000). 002 sería el campo tipo, el primer cero indica tipo de etiqueta universal, las de los tipos primitivos, segundo cero P/C=0 indica que es un tipo primitivo. El dos viene de la etiqueta 00010=2 de los números enteros. Cada tipo primitivo tiene una etiqueta propia, la de los enteros (INTEGER) es el número 2. El segundo campo 01 indica

que la longitud del campo datos sólo es de un byte, por último 00010000 es la codificación de 16 en complemento a dos, recordemos que los enteros también pueden ser números negativos y por lo tanto a la hora de codificarlos lo deberemos hacer de alguna forma.

Gracias a las etiquetas IMPLICIT se puede ahorrar el envío de gran cantidad de información

• Tipo Real. Como vimos en un ejemplo al principio del artículo, los reales necesitan tres números, mantisa exponente y base. BER permite la posibilidad de codificar los reales de diferentes maneras, sólo es necesario indicar qué manera se utiliza, esto se hace en los dos primeros bits del primer byte del campo datos. Si son dos ceros el número viene codificado según la norma ISO 6093, si es 1 el primer bit entonces se utiliza la forma binaria. Veamos en qué consiste esta última. Un primer byte, el primero del campo datos, estaría compuesto por el primer bit donde se indica que se trata de esta forma de codificación, el segundo el bit S que indica el signo de la mantisa 0 positivo y 1 negativo; a continuación la base expresada en dos bits B 00=2, 01=8, 10=16; después un factor de multiplicación F; por último, dos bits que indican el número de bytes del exponente según la siguiente manera 00=1, 01=1, 10=2, 11=3. El siguiente octeto indica el número de estos que ocupa el exponente. Los siguientes octetos son en primer lugar los del exponente E, indicados en el campo anterior y los siguientes un número N. Por lo tanto la expresión del número real será NR= S*N*2 ^ F*B ^ E, el valor absoluto de la mantisa sería N*2 ^ F

• Tipo BIT STRING. Este tipo, al igual que el integer resulta muy fácil de codificar, aunque presenta una particularidad, el campo tipo es igual que el integer, pero la etiqueta en vez de ser un dos es un tres. Lo especial aquí es que se añade un byte para indicar los bits de relleno del último byte. Un sencillo ejemplo sería (003 02 3 11101XXX), el dos indica que la longitud del campo datos es dos bytes y el tres, que tenemos tres bits de relleno.

Los tipos construidos seguirían un proceso recursivo. Los tipos CHOICE, como es lógico, sólo pueden incluir un campo de todos los posibles.

Por último veamos cómo modifican las etiquetas y la codificación de los tipos que las tienen puestas. Si tenemos puesta etiqueta, la indicamos en su campo, o si no ponemos la del tipo a la que se refiere, Integer 2, Boolena 1, Bit string 3 etc.

En el caso de que tengamos puesto el modificador *implicit*, nos ahorramos el poner información del tipo asociado y de su longitud, únicamente ponemos el dato.

La reglas de codificación BER no son lo que se dice eficientes pues necesitan mucha información para poder enviar los datos a mandar, a la vez que no soportan cifrado y compresión de los datos, lo que las hace manifiestamente mejorables.

Veamos a continuación un ejemplo que resume lo tratado y fija las ideas sobre ASN.1 y las VER:

```
EjemploFin ::= [ APPLICATION 1 ] IMPLICIT
     SET {
     valor [ 0 ] IMPLICIT INTEGER
     nombre [ 1 ] IMPLICIT IA5STRING
}
```

Una vez que tenemos definido el tipo pongamos un valor:

```
ejemplodato ::= EjemploFin {
     valor 12
     nombre "Fin del Articulo"
}
```

Para empezar a codificar nos fijamos en que tenemos una etiqueta application y que es *implicit* lo que nos obliga a no decir nada de set y de su longitud. Pero sí la longitud de EjemploFin. Tendríamos entonces: 111, el primer uno por ser la etiqueta tipo application, el segundo por ser tipo construido y el tercero por ser uno la etiqueta application. La longitud ahora no la sabemos, antes tenemos que codificar todo el dato. Del SET sabemos que valor es un entero con etiqueta private y con valor cero por lo tanto tampoco tenemos que poner nada del tipo entero. Le seguirían a los tres unos un tres por etiqueta private un cero por tipo primitivo y otro cero por el cero de la etiqueta después la longitud del campo valor y por último el valor 12. tendríamos 111 ?? 300 01 y 12. Haríamos ahora lo mismo con el campo nombre 301 16 "Fin del Artículo". Basta contar los bytes y poner su número en el campo longitud, que inicialmente desconocíamos.

111 21 300 01 "12" 301 16 "Fin del Artículo", después del 21 tenemos 21 bytes, cada carácter asci ocupa uno el campo tipo otro y el campo longitud otro.

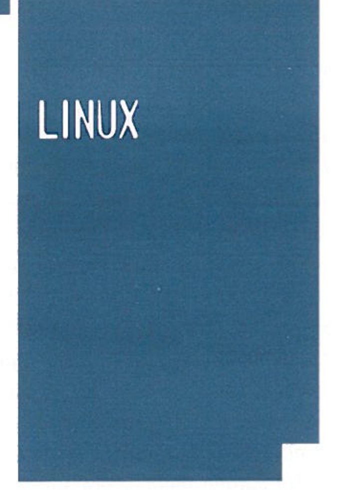
Conclusión

Como hemos visto a lo largo del artículo, ASN.1 logra que las aplicaciones distribuidas no sufran la dependencia del sistema operativo y de la arquitectura hardware, creando por debajo de ellas un lenguaje común, mediante el cual los datos, por muy complicados que sean, puedan ser interpretados correctamente. El complemento de ASN.1, que es BER, completa las reglas del lenguaje, que guardan una muy estrecha relación.

Bibliografía

• *Computer Network Third Edition,* Andrew S. Tanenbaum. Editorial Prentice Hall.

• *Protocolos de comunicaciones para sistemas abiertos,* José Miguel Alonso. Editorial Addison-Wesley Iberoamericana.



Programación X-Window: Trabajando con el acelerador gráfico

Jorge F. Delgado Mendoza

Con este artículo llegamos al final de nuestro viaje alrededor de la programación de tarjetas SVGA. Vamos a acabar esta serie de artículos estudiando algunos conceptos más avanzados de la programación de aceleradores, utilizando como ejemplo el Expansor de Color de nuestra tarjeta imaginaria Accelerated'r'Us.

Por fin llegamos al final de esta serie de artículos dedicada a la programación de drivers para tarjetas de vídeo dentro del entorno de XFree86, el sistema de ventanas gratuito para Linux basado en el X-Window System del M.I.T.

En el primero de los artículos describimos, de una manera muy general, la arquitectura de una Tarjeta de Vídeo SVGA. Para ello describimos sus componentes: DAC, frame buffer, chipset, etc., analizando las funciones que cada uno de ellos cumplía dentro del proceso de presentación de imágenes en la pantalla.

Fue en ese primer artículo donde se analizó en profundidad la estructura interna de un driver dentro de la familia de servidores de XFree86, enumerando los pasos que debíamos seguir para añadir soporte para una nueva tarjeta. Finalmente, seguimos el proceso descrito incluyendo el soporte básico SVGA - modos de 256 colores en resoluciones de hasta 1024x768 - para una tarjeta de nuestra invención: Graphics'r'Us.

En el segundo artículo profundizamos aún más en los diferentes aspectos de la programación de las características avanzadas de los chipsets, utilizando una segunda tarjeta imaginaria, Accelerated'r'Us, para ilustrar su implementación. En la segunda entrega aprendimos a utilizar un framebuffer lineal, añadimos más profundidad de color a los modos de vídeo y, finalmente, pudimos acceder a parámetros internos de configuración de la tarjeta a través del fichero de configuración del servidor.

En el número anterior dimos un gran salto cualitativo al comenzar a implementar funciones asociadas al acelerador hardware. Con la codificación del llamado cursor hardware aprendimos a utilizar varios ficheros a la hora de construir nuestro driver, así como a enlazar rutinas aceleradas con el bloque central del servidor. Esto último se realizó sustituyendo las funciones genéricas de la capa DIX (*Device Independent X*) del X-Window System por funciones dedicadas que aprovechasen al máximo las posibilidades que ofrecía la tarjeta.

A pesar de que el cursor hardware no sea una de las funciones más espectaculares en cuanto a mejora en el rendimiento, se decidió utilizarla como primer ejemplo por su claridad a la hora de realizar la implementación. Esta claridad reside, como se discutió en su día, en la correlación que existe entre el diseño hardware del cursor y las primitivas del servidor.

Sin embargo, en la mayoría de los aceleradores, la relación entre la implementación hardware y las primitivas de dibujado de la capa DIX del X-Window System no está tan clara. Por hacer un símil, la mayoría de las veces el acelerador nos proporciona unas cuantas toneladas de madera, mientras que lo que el

servidor espera es una mesa de caoba. El buen programador de drivers es capaz de aprovechar al máximo el potencial del acelerador, sustituyendo por rutinas aceleradas una gran parte de las funciones genéricas del servidor.

El aspecto más importante a la hora de decidir que funciones del servidor merece la pena implementar, cuáles no y en qué orden, viene dado por varios parámetros:

- **Frecuencia:** el parámetro más importante es la frecuencia de uso de una función determinada, definida como el número de accesos a esa primitiva partido por el número total de accesos a primitivas de dibujo por el servidor. Evidentemente, si aceleramos la velocidad con la que se ejecutan las funciones más frecuentes, el comportamiento del servidor mejorará radicalmente.

- **Cuellos de Botella:** es posible que una primitiva con un parámetro de frecuencia relativamente bajo, ocupe una gran cantidad de tiempo de ejecución generando un cuello de botella. Acelerando estas funciones conseguimos dos beneficios complementarios: por un lado, al hacerla más rápida mejoramos el rendimiento del entorno, mientras que por el otro la CPU queda libre para realizar trabajos del sistema. Este parámetro se caracteriza como el número de ciclos de reloj que necesita, en media, una primitiva determinada para dibujar un punto.

- **Grado un Mejora:** también es necesario tener en cuenta el grado en el que se mejora una función determinada. Desde el punto de vista subjetivo del usuario, no se empiezan a apreciar mejoras hasta que éstas alcanzan el 100%, mientras que para aplicaciones gráficamente intensivas, una variación del 20% puede ser muy significativa.

Evidentemente, las funciones por las que deberemos empezar serán las que tengan valores muy altos en los dos primeros parámetros, cuyos valores se pueden calcular con sistemas adecuados de perfilado del servidor. En cuanto al tercero de ellos, excepto en casos muy concretos, no es fácil calcularlo a priori, por lo que su peso será mucho menor a la hora de tomar la decisión de implementar una rutina acelerada.

Tabla 1. Valores de los diferentes parámetros de selección para algunas funciones del servidor del X-Window System.

	Frecuencia	Ciclos por Punto	Mejora
Rellenado Sólido	Alta	Medio/Bajo	Alta
Rellenado con Patrón	Media	Medio/Alto	Media
Scroll	Alta	Muy Alto	Baja
Transferencia de Bloques	Alta	Alto	Baja
Líneas Horizontales	Alta	Medio/Bajo	Alta
Líneas Verticales	Alta	Medio/Alto	Baja
Líneas Generalizadas	Media/Baja	Muy Alto	Muy Baja
Polígonos	Baja	Alto	Media/Baja

En la tabla 1 se muestran los valores que toman los parámetros de algunas funciones del servidor. El dato relativo al Grado de Mejora depende mucho del acelerador en sí y por tanto varía de una Tarjeta Gráfica a otra. Nosotros lo hemos calculado para el acelerador de nuestra última tarjeta Accelerated´r'Us.

Un rápido análisis de la tabla nos permite identificar dos candidatos potenciales a ser acelerados en primera aproximación: Rellenado Sólido y Líneas Horizontales. En ambos casos tenemos una función que se utiliza muy a menudo y cuyo grado de mejora es elevado dentro de las posibilidades de la tarjeta Accelerated'r'Us.

Nosotros nos vamos a decantar por la opción de Rellenado Sólido, ya que las líneas horizontales no son más que un caso extremo de rectángulo relleno en el cual la altura es la unidad. Para realizar la implementación de las funciones asociadas al Rellenado Sólido vamos a utilizar el Expansor de Color de la Tarjeta de Vídeo.

Requisitos

Como en todo proyecto software, es necesario establecer cuáles van a ser los medios, tanto materiales como de conocimientos, que se van a necesitar para realizar la implementación de las rutinas de rellenado sólido.

Conocimientos: el lenguaje de programación C es imprescindible a la hora de realizar código, ya que todo el servidor está realizado en este lenguaje.

Para el caso del rellenado sólido es, además, imprescindible un conocimiento profundo del ensamblador de la arquitectura x86 ya que, aunque las rutinas aceleradas se pueden realizar en lenguaje C, el ensamblador nos permite realizar las optimizaciones que nosotros creamos convenientes.

Desgraciadamente, cada Sistema Operativo,- Linux, NetBSD, FreeBSD, Solaris, etc..,- utiliza una sintaxis ensamblador diferente, por lo que tendremos que utilizar algún tipo de sistema para conseguir la compatibilidad entre todas las plataformas sobre las que ejecuta XFree86. Este artificio se denomina GNU Macroized Assembler, y su sintaxis se puede encontrar en el LinkKit, en el fichero: include/assintax.h

Si tan sólo necesitamos que el driver ejecute en Linux, podremos utilizar

las directivas __asm__ del GNU GCC, sin embargo es preferible utilizar el otro procedimiento ya que los beneficios que se consiguen, - portabilidad en todo el rango de Sistemas Operativos sobre los que ejecuta XFree86 - superan en gran medida al pequeño esfuerzo necesario.

Finalmente, deberemos tener una idea general de la temporización de las distintas instrucciones en los microprocesadores x86, así como las del chipset acelerado y las del bus a través del cual nos comunicamos con la tarjeta. Una idea clara de estos conceptos nos permitirá realizar una implementación óptima de las funciones aceleradas.

Software:

- gcc 2.7.0 o superior.
- make.
- XFree86.
- LinkKit.

Documentación: un manual del ensamblador del x86 con las temporizaciones de cada instrucción sería muy útil para sacar el máximo jugo al sistema. Un ejemplo clásico sería el de utilizar desplazamientos a la izquierda en lugar de productos si el multiplicador es una potencia de dos.

La documentación sobre el macro-ensamblador de GNU está en incluse/assintax.h

Si vamos a utilizar el macro-ensamblador de GNU, la documentación relativa a su sintaxis se puede encontrar, como ya hemos visto, en el fichero include/assintax.h, tomando como origen de coordenadas la raíz del LinkKit.

Por supuesto, deberemos tener a mano el manual de registros de la Tarjeta de Vídeo que vayamos a utilizar. En nuestro caso, el manual de la tarjeta imaginaria Accelerated'r'Us se puede encontrar en el CD-ROM de la revista en varios formatos: Texto, PostScript y Portable Document Format (.pdf).

Finalmente, el FAQ sobre tarjetas de vídeo SVGA tiene información detallada sobre la temporización de los buses, en distintas resoluciones y para un gran número de tarjetas. Además, el banco de pruebas de MS-DOS llamado VIDSPEED, o el programa de Linux denominado speedtest que viene incluido en la librería SVGALIB, nos dará una idea de la velocidad de la interfaz PCI de la tarjeta.

Expansor de Color: Qué es y Para Qué Sirve

La evolución del hardware destinado a presentar datos en pantalla ha sido paralela - en los últimos años podemos decir que incluso más rápida- a la evolución de los ordenadores compatibles PC.

En los últimos años hemos visto cómo las tarjetas que permitían 256 colores pasaban de ser el estándar a ser el mínimo imprescindible. Hoy en día, cualquier tarjeta es capaz de ofrecer modos de 16'7 millones de colores en resoluciones de 1024x768 y 65536 en modos de hasta 1280x1024 puntos.

Sin embargo, a pesar de que cada día la presentación de datos e imágenes en pantalla exige más y más colores, la gran mayoría de las aplicaciones destinadas a entornos de ventanas o GUIs (*Graphic User Interface*) utilizan únicamente dos colores a la vez en un momento determinado.

Basta pensar en la aplicación por excelencia, el Editor de Texto. Ya sea un editor de código o un complicado editor WYSIWYG, cuando estamos escribiendo las rutinas de pintado de caracteres sólo utilizan dos argumentos: el color de primer plano (las letras) y el color de fondo (la hoja).

Los Modos de Expansión de Color alivian el tráfico entre la Tarjeta y la CPU

Al analizar esta situación, los diseñadores de Tarjetas Gráficas llegaron a la conclusión de que, en modos de 256 colores, era un desperdicio auténtico enviar 8 bits por cada punto que se quisiera dibujar, cuando basta un '1' o un '0' para indicar si el punto es del color del primer plano o del color del fondo. En modos de más profundidad de color, la situación se hace aún más crítica, ya que cada punto pasa a ocupar 16, 24 o incluso 32 bits.

Como las transferencias a través del bus ha sido, durante mucho tiempo, el cuello de botella de las rutinas de dibujado, los diseñadores inventaron lo que se ha dado en llamar los Modos de Expansión de Color, para aliviar el tráfico a través del interfaz entre la Tarjeta y la CPU.

Cuando una tarjeta entra en modo de Expansión de Color, se guardan dos valores en sendos registros internos de la tarjeta. El primero de ellos guarda la entrada correspondiente al color de primer plano y el segundo la que representa al color de fondo. Una vez realizada esta tarea, cada byte que enviemos al frame buffer es expandido a 8 puntos diferentes. Aquellos que tenían un '1' se rellenan con el color de primer plano y los que estaban a '0' se dibujan con el color de fondo. Una vez haya terminado la función gráfica de dibujar, se reprograma la tarjeta para que conmute al modo normal.

De esta manera, hemos multiplicado el ancho de banda del bus por ocho, siempre y cuando ignoremos las pequeñas pérdidas debidas a la inicialización y reconfiguración de la tarjeta. Si la tarjeta expande además modos de 16 bits y 24 bits, las

ganancias son aun mayores. (Factores 16 y 24 ó 32 respectivamente).

Por poner un ejemplo real, una tarjeta de vídeo ISA poseía un ancho de banda de unos 3'5 Mbytes/s. Para dibujar el fondo del papel en una pantalla de 1024x768 en dieciséis bits, tardaría aproximadamente:

Información total transferida:
768x1024x2=1.5Mbyte

Tiempo total de transferencia:

$$\frac{1.5\text{Mbytes}}{3.5\text{Mbytes/s}} = 0.43\text{seg.}$$

Es decir, casi medio segundo ¡sólo para pintar el fondo!. Imagínate si además tenemos que poner las letras, los iconos, las barras, etc... Por supuesto, si queremos acceder a Internet y escuchar nuestro CD de música preferido mientras tanto, lo tenemos difícil. (O tenemos grandes dosis de paciencia).

Si utilizamos la expansión de color, las fórmulas quedan de la siguiente manera:

Información total transferida:

$$\frac{768\text{x}1024\text{x}2}{16} = 96\text{Kbytes} + 5\text{ bytes}$$

Los 5 bytes que hemos añadido se utilizan de la siguiente manera: 1 para configurar la tarjeta, para el color de primer plano y dos más para el color de fondo.

Tiempo total de transferencia:

$$\frac{96\text{ Kbytes}}{3.5\text{ Mbytes/s}} = 27\text{ mseg.}$$

Utilización relativa de la CPU del sistema:

$$\frac{0.027}{0,43} = 6,28\%$$

Es decir, no sólo hemos pintado el fondo mucho más rápido, sino que hemos liberado un 94% de la CPU para esas otras tareas que resultan imprescindibles en la vida de cualquier usuario. (Netsurfing, escuchar música, jugar al solitario, etc...).

Además de los registros de primer plano y de fondo, la mayoría de las veces nos encontramos otros dos elementos añadidos, cuya inclusión apenas supone un esfuerzo a los diseñadores hardware:

- **Registro Patrón:** en determinadas funciones como pueden ser el rellenado sólido, rellenado con patrones (tiles y stipples), etc.., se sabe de antemano qué secuencia de puntos se va a escribir en el frame buffer. Gracias a ello, es posible ahorrar ciclos de reloj introduciendo este dato en un registro interno de la tarjeta, de tal manera que podamos evitar tener que leer los valores de bus.

Figura 1. Registros asociados a un expansor de color de 8 bits perteneciente a la tarjeta imaginaria Accelerated'r'Us. Además de los registros típicos de control, primer plano y fondo, podemos encontrar los de patrón y máscara, así como un bit dedicado a controlar el orden en el que se presentan los puntos en pantalla. (Este tipo de detalles son muy útiles a la hora de realizar optimizaciones).

Definiciones de los Registros de la Tarjeta Accelerated'r'Us

REGISTROS 0x10 - 0x1F: UNIDAD DE EXPANSIÓN DE COLOR

3DF — **Expansor de Color** — **Índice = 10** — **R/W**

Bit	Descripción
0	Selector de Expansión
	Bit 0 / Descripción: 0 — Modo de funcionamiento normal; 1 — Expansión de Color activada. Los datos escritos a la memoria de vídeo dependen del contenido de los registros 0x11, 0x12 y x13.
1	Selector de Datos
	Bit 1 / Descripción: 0 — Los valores del *BUS* de datos de la CPU son utilizados para seleccionar colores de fondo o primer plano; 1 — Los colores de fondo y primer plano se seleccionan en función de los contenidos del registro 0x13.
2	Enmascaramiento
	Bit 2 / Descripción: 0 — Se utiliza el sistema VGA de enmascaramiento; 1 — Enmascaramiento según el registro 0x14.
3	Orden de la Máscara
	Bit 3 / Descripción: 0 — Bit 7 se corresponde con el pixel 7 de un carácter (msb); 1 — Bit 7 se corresponde con el pixel 0 de un caracter (lsb)
7-4	Reservados.

Por Defecto: 0x0

3DF — **Primer Plano** — **Índice = 11** — **R/W**

Bit	Descripción
7-0	Estos 8 bits definen el color de primer plano cuando se habilita la expansión de color.

3DF — **Fondo** — **Índice = 12** — **R/W**

Bit	Descripción
7-0	Estos 8 bits definen el color de fondo cuando se habilita la expansión de color.

3DF — **Formato de la Expansion** — **Índice = 13** — **R/W**

Bit	Descripción
7-0	Estos bits definen cual de los colores se va a utilizar para cada punto. Si el valor es '1', se utiliza el contenido de *Primer Plano*. En caso contrario se utiliza el valor que se encuentra en *Fondo*

3DF — **Registro de Enmascaramiento** — **Índice = 14** — **R/W**

Bit	Descripción
7-0	Máscara.
	Bit 7-0 Descripción: 0 — Enmascara los datos; 1 — Escribe los datos.

Sólo Programadores Tech. Inc.

- **Registro de Máscara:** existen también funciones, esencialmente en creación de texto, que lo que quieren es dibujar encima de algo ya existente. Para evitar engorrosas tareas de cálculo de bordes, los expansores suelen tener incorporado un registro que permite enmascarar los valores que provienen del bus decidiendo si pasan o no al frame buffer.

Una vez sabemos qué es un Expansor de Color, ha llegado el momento de decidir qué funciones del servidor vamos a sustituir y, sobre todo, cómo vamos a utilizar el hardware para implementarlas.

Especificación de Requisitos

A la hora de programar el acelerador de la Tarjeta Gráfica, conviene que comencemos por una rutina fácil y sencilla que nos ayude a fijar los conceptos de una manera clara.

Para ello vamos a implementar una de las rutinas más utilizadas del servidor del X-Window System: el Rellenado Sólido, sin máscara y utilizando GXCopy como ROP . Implementando esta función substituiremos por funciones aceleradas el 85% de los accesos a primitivas de rellenado sólido del servidor.

Fijándonos en otros drivers y en los fuentes del servidor, podemos encontrar los puntos de entrada de las funciones básicas para tarjetas VGA compatibles. Estos puntos de entrada, o primitivas gráficas, están agrupados en una estructura de funciones denominada:
vga256LowlevFuncs

De entre todas las funciones disponibles, dos llaman nuestra atención por sus nombres:

vga256LowlevFuncs.fillBoxSolid
vga256LowlevFuncs.fillRectSolidCopy

Los nombres de las funciones que el servidor asigna por defecto a ambos campos de la estructura son:

vga256FillBoxSolid
vga256FillRectSolidCopy

Un análisis en profundidad del código asignado a las mismas, nos permitirá identificar a la primera de ellas como la encargada de realizar todas las operaciones de rellenado sólido del servidor. La segunda es un caso particular de la primera, en la cual tenemos fijados ciertos parámetros.

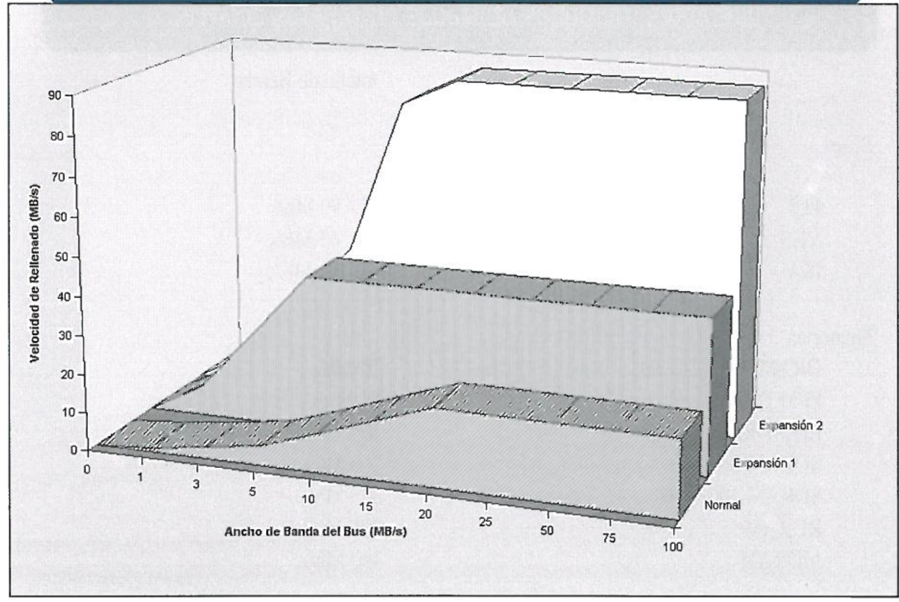
Nuestras funciones aceleradas deberán, por tanto, sustituir exactamente a este par de funciones, realizando la misma tarea que éstas tienen encomendada. Deberemos pues, realizar un par de funciones que dibujen conjuntos genéricos de rectángulos, utilizando el acelerador cuando éstos cumplan determinadas condiciones, - sean sólidos y con la ROP GXCopy - y llamando a la función software del servidor cuando no lo hagan.

Finalmente, he de hacer notar que sólo vamos a implementar estas funciones en el caso de tener habilitado el direccionamiento lineal. El motivo es conseguir claridad en la elaboración del código, ya que si implementamos las funciones para direccionamiento segmentado, tendremos que añadir una batería de funciones de cambio de banco, lo cual no aporta nada nuevo y sí ayuda a complicar el código resultante.

¿Merece la Pena Realizar la Implementación?

Una vez hemos decidido qué es lo que queremos hacer, debemos estudiar si merece la pena realizarlo. Para ello tendremos que analizar si de verdad se va a producir una mejora sustancial en el

Figura 2. Comparativa entre las velocidades de rellenado de una tarjeta de vídeo VLB con un ancho de banda de bus Bth de 20 MBytes/seg. Los valores correspondientes a la leyenda Expansión 1, suponen un ancho de banda en el frame buffer de 40 MBytes/seg correspondiente a una tarjeta de la primera generación. Los valores correspondientes a Expansión 2 corresponden a un interfaz de memoria más moderno, capaz de alcanzar los 87 MBytes/seg. Como se puede observar, la mejora en prestaciones debida a aceleraciones del tipo expansor es mucho más acusada en arquitecturas cuyo cuello de botella sea el bus de acceso.



rendimiento del servidor al introducir nuestras nuevas funciones aceleradas. Si no es así, volveremos a la fase de identificación de cuellos de botella para elegir una nueva candidata.

Hay que analizar si se va a producir una mejora sustancial en el rendimiento del servidor con nuestras funciones aceleradas

En el caso de la función que nos ocupa, suponemos que el factor limitador de la velocidad de nuestra tarjeta, a la hora de rellenar un rectángulo con un color determinado, es el ancho de banda del bus situado entre la tarjeta y la CPU. Para ello, la CPU transmite un único bit por punto en lugar de los ocho que le corresponderían. Una vez llegan a la tarjeta, el acelerador expande los unos al color de primer plano y los ceros al color de fondo, escribiendo los puntos completos en la memoria de vídeo.

Tabla 3. Resultados de las pruebas realizadas sobre tarjetas de vídeo Oak OTI087 con el programa xbench. Los resultados demuestran como a pesar de la mejora en la velocidad de la primitiva, el ancho de banda del frame buffer actúa como limitador del rendimiento del sistema. La resolución utilizada fue de 1024x768 8bpp.

	Sin Expansión	Con Expansión	Factor
OTI087 ISA	3.5 Mb/s	12.8 Mb/s	3.65
OTI087 VLB	11.7 Mb/s	21.3 Mb/s	1.8

Al realizar este proceso, y siempre dependiendo de la relación existente entre el ancho de banda del bus (Bth) y el ancho de banda de la memoria de vídeo o frame buffer (FBth) pueden darse tres casos:

- **Bth << Fbth** : En este caso, la expansión de color es muy útil, ya que va a multiplicar, hasta por ocho, la velocidad con la que se dibujan los rectángulos rellenos. Es el caso ideal, que se da en antiguas tarjetas ISA con memorias convencionales o en modernas tarjetas PCI con memorias de última generación.
- **Bth ~ Fbth** : Aquí, la diferencia de velocidades no va a ser realmente apreciable, sin embargo, vamos a disminuir la congestión en el bus, liberando ciclos de CPU para otras tareas. Este motivo hace que, aunque el subsistema gráfico no vaya a experimentar ninguna mejoría, sea aconsejable implementar la función.
- **Bth >> Fbth** : Si la situación es ésta, entonces nos hemos equivocado en la elección de aceleraciones, ya que el cuello de botella no es el bus de acceso, sino la memoria de la Tarjeta Gráfica. Implementar rutinas que estén basadas en expansión de color no va a suponer ventaja alguna sobre los métodos software más convencionales.

Debemos recordar que el ancho de banda disponible en el frame buffer depende, en tarjetas sin VRAM, de la resolución y el número de colores que se estén utilizando. Es posible que en algunos modos merezca la pena usar expansiones, mientras que en otros no sea aconsejable.

En cuanto a la mejora aparente se refiere; de cara al usuario apenas si se notaba diferencia en el caso de la tarjeta VLB, mientras que la apreciación subjetiva de mejora en el caso de la tarjeta ISA ofrecía un resultado muy espectacular.

Tabla 2. Anchos de banda para distintos buses de expansión y diferentes memorias utilizadas en Tarjetas de Vídeo.

	Ancho de Banda
Buses	
PCI	15 - 99 Mb/s
VLB	8 - 45 Mb/s
ISA	3 - 5 Mb/s
Memorias	
DRAM	20 Mb/s
FPM DRAM	87 Mb/s
EDO DRAM	105 Mb/s
SDRAM 66Mhz	166 Mb/s
SDRAM 100 MHz	253 Mb/s
RDRAM	206 Mb/s
MDRAM	490 Mb/s

Compliquémonos la Vida: Ensamblador

Si programar en lenguaje C es muchas veces engorroso,- sobre todo programando a bajo nivel - incómodo y difícil de mantener, a la mayoría de los programadores se les cae el alma a los pies simplemente escuchando la palabra ensamblador.

Los contras de programar en ensamblador son numerosos; la legibilidad del código brilla... por su ausencia, es imposible de mantener, (ni siquiera por el mismo que lo programó si ha pasado suficiente tiempo), etc., etc., etc...

Sin embargo, existen casos en los que es imprescindible realizar una vuelta a los orígenes y descubrir que el ensamblador también tiene sus usos. El más típico de ellos es la programación optimizada de dispositivos hardware de los que se tiene un profundo conocimiento. El kernel de Linux tiene alguna que otra línea de ensamblador, típicamente en puntos críticos del código.

Nosotros no vamos a ser menos y, aunque es posible realizarlo en C y dejar que el gcc realice todas las optimizaciones del mundo, utilizaremos ensamblador para realizar el bucle interno de las subrutinas de rellenado sólido. La razón es muy sencilla: la experiencia nos demuestra que, bien utilizado, el lenguaje máquina es sensiblemente más rápido.

En el caso del código de las tarjetas basadas en el OTI087 de Oak Technologies Inc, la misma rutina, con el mismo pseudocódigo fue realizada primero en C y después en ensamblador. Los resultados fueron muy ilustrativos: mientras que la rutina en C apenas llegaba a los 14 Mb/s, el código ensamblador ofrecía un rendimiento muy superior con más de 21 Mb/s.

Existe además otro motivo para utilizar ensamblador en el ejemplo de driver para nuestro chipset Accelerated'r'Us. Vamos a familiarizarnos con la sintaxis del macro- ensamblador de GNU, el cual puede ser muy útil si queremos seguir profundizando en la escritura de drivers para dispositivos.

Para concluir, escribiremos el código C de los enganches con el servidor del X-Window System y las líneas de ensamblador necesarias para extraer la máxima velocidad del acelerador de nuestra tarjeta.

Rutinas Aceleradas

Vamos a comenzar a realizar la implementación de las rutinas necesarias para obtener aceleraciones hardware.

Lo primero que haremos será preparar los ficheros necesarios para compilar el driver. Tocaremos el Imakefile para añadir los nuevos componentes de nuestro programa y crearemos los ficheros que vayan a contener el código asociado al Expansor de Color: uno para el código C y el otro para el código ensamblador.

Una vez realizada esta tarea, utilizaremos la función FbInit() de nuestro driver para informar al servidor de la presencia de las nuevas primitivas de dibujo. Es en FbInit() donde deberemos decidir si las características del servidor nos permiten utilizar las aceleraciones hardware o no.

Preparando el Driver

Lo primero que tenemos que hacer es incluir el nuevo fichero en el Imakefile de nuestro driver. Para ello, procederemos de manera similar al caso del cursor hardware, añadiendo una entrada para el nuevo fichero gu_colexp.c en el Imakefile.

En FbInit() decidimos si las características del servidor nos permiten utilizar las aceleraciones hardware o no

A continuación crearemos el fichero gu_colexp.c, añadiendo los #include

Figura 3. Lista de las inclusiones necesarias para la implementación de sustitutos acelerados de las funciones de Rellenado Sólido del servidor de XFree86.

```
#include "cfbrrop.h"
#include "stdio.h"
#include "vga256.h"
#include "mergerop.h"
#include "vgaBank.h"
#include "xf86.h"
#include "vga.h"                /* Para vgaBase. */
#include "compiler.h"
#include "gu_driver.h"
#include "xf86_Config.h"
```

necesarios para poder implementar nuestras funciones de rellenado sólido.

Una vez realizado este proceso, lo repetiremos para el fichero donde vamos a implementar las rutinas en ensamblador. Si para incluir gu_colexp.c en el Imakefile, nos fijamos en gu_driver.c, para incluir gu_colexp.s usaremos como modelo el fichero de cambio de banco gu_bank.s

Sustituyendo las Funciones del Servidor

Para localizar las funciones que corresponden a cada primitiva de dibujo debemos mirar en varios lugares del árbol de fuentes del servidor. Los lugares típicos donde podemos encontrar código útil para nuestros propósitos son:

- **CFB**: acrónimo de *Color Frame Buffer*. En él podemos encontrar todas las funciones software genéricas que se utilizan en el servidor. Si buscamos funciones para modos de 16 y 32 bits, deberemos mirar en cfb16 y cfb32 respectivamente.
- **MI:** una acrónimo también, en este caso de *Machine Independent*. Dentro de la familia MI de funciones podemos encontrar un montón de ejemplos independientes de la arquitectura. Son funciones menos optimizadas que las de cfb.
- **VGA256:** en esta familia de funciones podemos encontrar código relativo a tarjetas VGA que utilicen modos de 256 colores.
- **Otros Drivers:** también podemos encontrar ejemplos interesantes en el código acelerado para otras tarjetas de vídeo. Un ejemplo muy completo podría ser el presente en el directorio de las tarjetas Cirrus Logic.

Como ya vimos en el apartado de Especificación de Requisitos, las funciones que debemos sustituir son:

```
vga256FillBoxSolid
vga256FillRectSolidCopy
```

las cuales se enganchan al servidor en la estructura:

```
vga256LowlevFuncs
```

Los componentes de esta estructura a los que debemos enganchar nuestras funciones aceleradas son:

```
fillBoxSolid
fillRectSolidCopy
```

Para completar la sustitución realizaremos las siguientes tareas:

- Añadiremos los #include necesarios en el servidor, de tal manera que tengamos acceso a todas las estructuras de datos pertinentes.
- Inicializaremos dos arrays de datos que nos servirán más adelante para enmascarar los datos de los bordes de los rectángulos que dibujemos.
- Comprobaremos entonces que vamos a emplear direccionamiento lineal para los diferentes modos del servidor.
- Si la tarea anterior dio un resultado positivo, realizaremos la sustitución mediante las siguientes líneas de código:

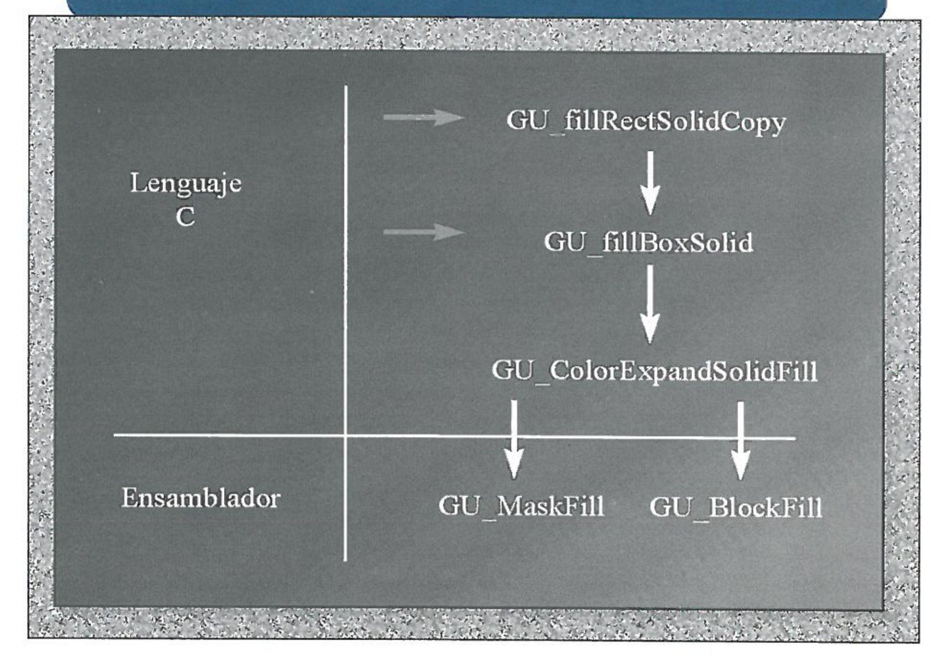
```
vga256LowlevFuncs.fillBoxSolid =
        GU_fillBoxSolid ;
vga256LowlevFuncs.fillRectSolidCopy =
        GU_fillRectSolidCopy;
```

Una vez realizados todos los puntos anteriores, únicamente nos queda implementar las funciones aceleradas siguientes:

```
GU_fillBoxSolid
GU_fillRectSolidCopy
```

Una vez realizadas, únicamente nos quedará compilar el driver, generar un servidor y realizar las pruebas necesarias para comprobar que funciona correctamente. Los pasos necesarios para realizar estas operaciones se describieron en detalle en los artículos I y II de la serie.

Figura 4. Dependencia funcional de los diferentes componentes del código acelerado. Las flechas oscuras indican puntos de acceso del servidor del X-Window System.



El Código

La primera función a implementar es GU_fillRectSolidCopy, ya que nos vamos a limitar a reorganizar los parámetros de entrada de la misma y llamar a GU_fillBoxSolid con unos valores procedentes de la reorganización antes comentada.

A continuación realizaremos GU_fillBoxSolid, para lo cual comprobaremos que los parámetros de entrada se corresponden con los de Rellenado Sólido GXCopy. Si es así, llamaremos a la función GU_ColorExpandSolidFill, la cual se encargará de hacer la interfaz con nuestras funciones en ensamblador. Si la llamada a GU_fillBoxSolid no se corresponde con uno de los casos para los cuales hemos realizado la implementación de aceleraciones, utilizaremos las funciones genéricas del servidor.

Existe un caso especial en el cual, a pesar de ser rellenado GXCopy, no vamos a utilizar nuestras rutinas aceleradas. Si el ancho de la zona a rellenar es menor de diecisiete puntos, el proceso de activar e inicializar el expansor es más lento que realizar la función por software. Por tanto, en estos casos utilizaremos las funciones del servidor.

La función intermedia GU_ColorExpandSolidFill, manipula los datos de entrada, activa el expansor y divide el área a dibujar en tres partes. A continuación llama a las dos funciones en ensamblador: GU_BlockFill y GU_MaskFill. Finalmente, se deshabilita la Expansión de Color y se devuelve el control al servidor.

La división en tres partes de la zona a dibujar nos viene forzada por la imposibilidad de dibujar puntos individuales cuando activamos el modo de expansión. Esta particularidad hace que, para pintar rectángulos cuyos bordes no estén alineados con bytes enteros en el frame buffer, tengamos que utilizar las máscaras disponibles a través del registro 0x14 de la tarjeta. Las tres partes en la que dividimos el área a dibujar son las siguientes:

- **Parte Central:** comprende todas las columnas de la memoria en las cuales dibujamos el byte de datos entero.
- **Parte Izquierda:** si existe, es debido a que la primera columna del rectángulo no está alineada con la dirección de un byte. Al dibujarla utilizaremos el registro de máscara para evitar salirnos del área de dibujo.
- **Parte Derecha:** de nuevo, puede no existir. La última columna del rectángulo tampoco tiene porqué estar alineada con una frontera de byte. Utilizamos el registro de enmascaramiento de manera análoga al caso anterior.

La división en tres partes se produce por no poder dibujar puntos individuales al activar el modo de expansión

Para concluir, implementamos las funciones en ensamblador, las cuales se limitan a rellenar un rectángulo de varios bytes de ancho - GU_BlockFill - o bien una columna de un único byte de ancho - GU_MaskFill -.

En ambos casos utilizamos un argumento denominado pitch, el cual contiene el valor de la Anchura Virtual de la pantalla. Este valor es imprescindible ya que la función debe funcionar con cualquier resolución virtual, lo cual hace que no sepamos a priori el número de puntos que tenemos que sumar a una dirección para colocarnos en la línea siguiente.

Una vez realizadas todas estas funciones, llegó el momento de instalar y probar el driver. Si queremos sorprendernos con la gran mejora que habrá experimentado el funcionamiento del servidor, podemos utilizar el programa xbench, una copia del cual se puede encontrar en el CD de la revista.

Conclusiones

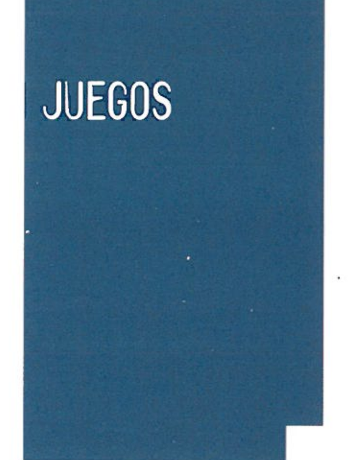
Por fin hemos llegado al final de esta serie de cuatro artículos sobre programación de drivers SVGA para el X-Window System de XFree86. Aunque parezca mentira, lo que comenzó como una mera introducción a la programación de Tarjetas de Vídeo que nos permitiera utilizar el GUI de Linux si nuestro chipset no estaba soportado, se ha acabado convirtiendo en una sólida base que nos va a permitir enfrentarnos a cualquier tipo de acelerador gráfico.

Por supuesto que hay mucho más. Es imposible enseñar a programar drivers de calidad profesional en las treinta y nosecuantas páginas que hemos dedicado al tema. Sin embargo, con los conocimientos que hemos adquirido y un poco de trabajo, podremos enfrentarnos sin miedo a cualquier tarjeta que se nos presente - siempre y cuando tengamos la documentación apropiada.

En cuanto a añadir más funciones aceleradas, el procedimiento es fácil. Localiza un punto crítico en el rendimiento, aisla la función culpable en el servidor y sustitúyela por una variante hardware de la misma adaptando las posibilidades del acelerador a esa tarea. Así se hace para implementar todo tipo de soluciones, desde rellenado sólido hasta bilinear filtering.

Y, para concluir, recuerda: ¡Nunca debes llevar al hardware más allá de sus especificaciones... a no ser que sepas lo que estas haciendo!

Jorge Delgado Mendoza es miembro de XFree86, como desarrollador responsable del servidor de Oak Technologies Inc. Es Ingeniero Superior de Telecomunicaciones por la UPM y trabaja en Convex Supercomputers. Su dirección de correo es ernar@dit.upm.es



Movimiento selectivo. Extensores

Eugenio Castillo Jiménez

La respuesta generada por un programa de ajedrez está basada en la búsqueda a través de un árbol de jugadas que se expande por fases de profundidad (pliegues) cuyo número depende del nivel de juego seleccionado por el usuario.

Todo análisis producido dentro de dicha profundidad (profundidad nominal, PN a partir de ahora) se corresponde con la llamada fuerza bruta, donde el algoritmo del minimax y la simplificación del alfa-beta determinan la forma del árbol. Con el sistema de búsqueda de tranquilidad o de capturas (primer paso hacia el movimiento selectivo, se consideran únicamente las capturas y se renueva el alfa-beta al iniciar cada nodo y no cuando se devuelven valores) logramos ampliar el árbol más allá de PN en lugares concretos.

Extensores

En el ajedrez y en otros juegos la mayor parte de las situaciones que se presentan sólo exigen respuestas que reduzcan al máximo el espacio de posibilidades dominado por el rival y se dan pocas ocasiones en las que existan jugadas ganadoras, denominaremos de este modo a aquellas que concedan una ventaja suficiente como para dar garantías de victoria, la ventaja proporcionada siempre suele ser de dos tipos: material o conceptual. Un programa de ajedrez es consciente de la primera porque la ha generado dentro del árbol y de la segunda porque su sistema de evaluación es capaz de identificar el concepto decisivo.

La mayor parte de las jugadas ganadoras no identificadas se corresponden con las de ventaja material (ya sea mediante capturas o por jaques al rey), en estos casos no se trata de simples combinaciones que alcanzan la ventaja material de un modo inmediato, más bien requieren de un sacrificio (ofrecimiento de material). Si el punto de inflexión (momento en que se adquiere la ventaja) cae fuera de PN (coto de la fuerza bruta) el sistema de búsqueda de tranquilidad (o de capturas) suele fallar porque la secuencia comienza con desventaja.

Todo análisis producido dentro de la profundidad nominal se corresponde con la llamada fuerza bruta

Intentando resolver esta carencia un gran número de programadores de ajedrez se han puesto manos a la obra para expandir (incrementar la PN) el árbol de jugadas justo allí donde es necesario o donde la expansión parece tener sentido, tratando a su vez de aminorar al máximo el efecto de la explosión de posibilidades que ello conlleva. Mediante la experiencia y los múltiples estudios estadísticos se han identificado una serie de jugadas clave que justifican un incremento en la PN haciendo más fuerte el juego de un programa. Veamos algunas de las más importantes:

1. **Recaptura:** Si una pieza que captura a otra es a su vez capturada se produce una recaptura. Por lo general estas jugadas producen un efecto de cuello de botella en el árbol de nodos debido a que las respuestas lógicamente válidas después de una captura se reducen a la recaptura en un alto número de casos. Esto equivaldría a un decremento de la PN, si producimos ahora un incremento (extensión) artificial tendemos a compensar este efecto y a su vez hacemos más exacta la respuesta.

Para estos casos algunos programadores dan un incremento automático de la PN, otros comparan previamente el valor de la posición con el alfa en curso, sólo si el valor cae por encima se produce el incremento. Ambos sistemas poseen sus pros y contras. El primero dará unas respuestas más exactas pero provocará en algunos casos estallidos de probabilidades que tenderán a ralentizar la búsqueda, el segundo reduce el número de incrementos por recapturas a los exclusivamente lógicos (empleando una lógica materialista) lo cual no siempre es cierto, además para casos de reevaluaciones si el alfa es muy alto puede que en el siguiente pliegue el incremento no se produzca debido al alto valor del alfa y con ello tampoco se identifique la jugada clave que en un ply anterior sí se vio.

2. **Jaques:** También aquí se presentan los cuellos de botella aunque esta vez sea por motivos de reglas, en este caso las jugadas legales se reducen a una quinta o sexta parte.

3. **Peón pasado avanzado:** El último movimiento se corresponde a un peón pasado avanzado. La importancia de estos movimientos es vital y muchas veces sus terribles efectos no son prevenidos con antelación por falta de profundidad, si no se aplica la extensión no se suele ver el siguiente avance que es matador, o el que una pieza contraria lo obstaculice haciéndolo inválido, o el que una torre lo pueda apoyar haciéndolo mortífero. En la mayor parte de los casos estas extensiones sí que provocan explosiones de posibilidades, pero la información obtenida merece la pena.

4. **Respuesta unitaria:** El CRAFTY incorpora esta original idea para los casos en los que exista una sola respuesta válida. Estas situaciones se pueden dar en finales de partidas o casos de jaque.

5. **Factor pieza colgante/hung:** El GNU considera este factor como activo en las posiciones en que el bando que mueve tiene serias posibilidades de perder una pieza. Se supone que ello puede ocurrir cuando una pieza es amenazada y no tiene una casilla de huida clara, o cuando más de una pieza se encuentra amenazada (como sólo podemos mover una la otra podría quedar "colgada").

6. **Huida:** Cuando la jugada la realiza una pieza que puede ser capturada favorablemente por el contrario. Muy pocos programadores utilizan este posible extensor. Su efectividad no ha podido ser demostrada pero con ciertas modificaciones o ampliaciones puede resultar de interés.

Podemos observar cómo se implementan en C algunas estas ideas en la tabla 1.

El ejemplo de la tabla 1 es una muestra del uso básico de extensores, la primera comparación referida al chequeo de jaques aumenta depth en uno si nos encontramos a menos de dos plys del final de la PN, siguiendo esta regla ello implica que en condiciones normales nunca se producirá más de una extensión por jaques en una misma línea, lo cual es a todas luces insuficiente. Las dos siguientes comprobaciones se refieren a los peones pasados avanzados y a las recapturas. En este caso el código de recapturas no posee la condición "..&& score > alpha &&.." que prefieren algunos programadores.

El CRAFTY usa el array :

```
typedef enum {no_extension=0,
      check_extension=1,
      recapture_extension=2,
      passed_pawn_extension=4,
      one_reply_extension=8}
EXTENSIONS;
EXTENSIONS
 extended_reason[MAXPLY];
```

para señalar los ply con extensores y el motivo de su extensión. Ello le permite

Tabla 1

```
//depth  : el número de pliegues que quedan por analizar .
//ply      : nivel del pliegue en curso.
//Sdepth : límite inicial del número de pliegues en el CRAFTY equivale al iteration_depth

//**ejemplo extensores GNU**

if (depth > 0) {
  if (InChk) //InChk es una variable booleana que indica si el rey del bando que mueve está en
        //jaque
    depth = (depth < 2) ? 2 : depth;
  else if ((ply>1 && PawnThreat[ply - 1]) || /*comprobación si se ha movido un peón más allá
        de la 6º fila en el ply anterior*/
        (ply>2 && CptrFlag[ply - 1] && CptrFlag[ply - 2] &&
         ply<Sdepth+2 && CptrFlag[ply-1]==CptrFlag[ply-2])) //Chequeo de las recapturas
       ++depth;
}
```

crear una limitación en los extensores por recapturas, y es que no deja que se produzcan dos extensiones por recaptura seguidos usando "!(extended_reason[ply-1]&recapture_extension)".

Veamos algunos ejemplos extraídos del CRAFTY:

```
//Ejemplo para recapturas

if (!extended_reason[ply]) {
//chequea que no se hayan dado extensiones
//previas en el mismo ply
      if (wtm)
//wtm = white to move : mueven blancas
      new_mev=Material(ply)+
piece_values[Captured(current_move[ply])]+
 piece_values[Promote(current_move[ply])];
  else
     new_mev=Material(ply) –
 piece_values[Captured(current_move[ply])]-

        piece_values[Promote(current_move[ply])];
    if (recapture_extensions && ((ply+depth) <=
        2*iteration_depth) &&
        Captured(current_move[ply-1]) &&
        Captured(current_move[ply]) &&
        (To(current_move[ply-1]) ==
        To(current_move[ply])) &&
        (Material(ply-1) == new_mev) &&
        !(extended_reason[ply-1]
              &recapture_extension))
    {
      extended_reason[ply]|=recapture_extension;
      recapture_extensions_done++;
      tdepth++;
    }
}
```

En este código new_mev es una variable que recoge el estado material del tablero después de la jugada, CRAFTY comprueba que la recaptura además haya producido un reequilibrio material así por ejemplo una secuencia "peón captura caballo y alfil captura peón" no daría una extensión por existir una desventaja material.

Obsérvese que en los dos ejemplos de recapturas ambos colocan un límite al número de extensiones dependiendo del grado inicial de profundidad de búsqueda establecido (CRAFTY : "(ply+depth) <= 2*iteration_depth)"; GNU: "ply < Sdepth + 2").

```
//ejemplo para jaques CRAFTY

# define FutileBehind(wtm,ply) ((wtm) ?
        Material(ply) <= -ROOK_VALUE : \
        Material(ply) >= ROOK_VALUE)

# define Check(ply,wtm)
        \
   Attacked((wtm) ? position[ply].white_king :
        \
        position[ply].black_king,ply,!wtm)
..........

if (Check(ply,wtm)) {
   if (ply > 1) {
     in_check[ply]=1;
     if (check_extensions &&
!FutileBehind(wtm,ply) &&
((ply+depth) <= 2*iteration_depth))
{
   extended_reason[ply]=check_extension;
   check_extensions_done++;
   depth++;
   }
  }
 }
```

En este ejemplo FutileBehind es un chequeo de la cantidad de material existente, por si se busca un mate imposible por falta de piezas, Check sólo comprueba si está en jaque el rey.

Uno de los primeros casos de ampliación en los extensores surge precisamente en los jaques, si nos encontramos con un jaque fuera de PN (depth < 1) los extensores no actúan. Podemos/debemos hacer una excepción para este caso, si el alfa es menor que valorinicial entonces convendría realizar un chequeo de posición de peligro del rey para determinar si sería conveniente transformar este ply de depth 0 en depth 1. El poner como condición el "alfa < valorinicial" es relativo pero en un gran número de casos cierto, la idea de realizar la ampliación de PN es en este caso sólo para perjudicar (siempre que se demuestre en el árbol) al bando que está en jaque y si "valorinicial < alfa" ello significa que ya existe una variante superior a la analizada, y si ésta ya es desechable por inferior ¿para qué perder el tiempo en demostrar que es aún más inferior?

Veamos la codificación de este caso específico (en el ejemplo extensores GNU) sustituyendo la línea de "if (InChk)":

```
if (InChk)
{ f = Ocupadas (side);
  if (f > 4)
  { ++depth;
    if (Sdepth + 4 < ply)
    { if (f < 7)
        --depth;
    }
  }
}
else ...........

//Ocupaciones de jaques

int Ocupadas (short int side)

{ short u,res,sq;
 unsigned char far *pdir;
 if (side == white)
 { sq = PieceList [white][0]; res = reyarea [sq];
  pdir = nextpos+king*64*64+sq*64;
  u = pdir[sq];
  do
  { if ((atak[black][u] > 0) || (color [u] == white))
      res++;
    u = pdir[u];
  } while (u != sq);
  return res;
 }
 else
 { sq = PieceList [black][0]; res = reyarea [sq];
  pdir = nextpos+king*64*64+sq*64;
  u = pdir[sq];
  do
  { if ((atak[white][u] > 0) || (color [u] ==
      black))
     res++;
     u = pdir[u];
  } while (u != sq);
  return res;
 }
}
```

reyarea es un array precalculado que da el número de casillas que no puede ocupar

un rey por estar próximo a los bordes del tablero, el caso de una esquina es el peor con 5, un caso de centro de tablero sería de 0. res se va incrementado conforme va topando con casillas atacadas por el bando contrario o por casillas ocupadas por piezas propias (no se puede comer una pieza propia).

Siempre que Ocupadas nos de un valor superior a 4 el caso de jaque puede ser de mate y se amplia PN (o se transforma el ply en equivalente), si llevamos ya un cierto tiempo dando jaques sin conseguir nada ("if (Sdepth + 4 < ply)") es preciso afinar y obligar al programa a considerar como ampliables sólo los casos de jaques con sólo una casilla de escape para el rey ("if (f < 7) --depth"). Estas condiciones no abarcan un buen número de casos (por ejemplo un jaque de dama con 3 casillas de escape para el rey que termina en mate) pero nos resuelven satisfactoriamente un 70% de los problemas de mate, las posibilidades de ampliación son evidentes y no excesivamente complicadas.

Hay que señalar que después de un turno en el que el rival ha realizado un incremento de "depth" por motivos de jaque si el bando rival (jaqueador) está fuera de PN no generará todas jugadas que persigan al rey, para ello será preciso construir un algoritmo especializado que nos incluya a su vez este tipo de movimientos.

Vemos un ejemplo con un algoritmo de este tipo en la tabla 2.

jaqpub contiene un tablero generado con anterioridad de tal manera que cada escaque es un byte que indica si una pieza del tipo a chequear puede dar jaque desde esa posición. jaqalf, jaqdam y jaqtorr contienen el posible valor de ese jaque basándose en la cantidad de casillas que le quedarían al rey para moverse después del jaque, la variable signal activada cuando ascendemos en el árbol (en este caso tres pliegues por encima de PN) y ya sólo considera los jaques más peligrosos evitando de este modo jugadas innecesarias. Veamos ahora un ejemplo "sobre el tablero" acerca de la utilidad de los extensores (ver figura 1):

Tabla 2. Esta versión del capture está especialmente diseñada para los jaques

```
void
CaptureListJaq (short int side, short int ply)
/* Llena el array Tree con jaques,capturas y promociones del bando en juego. En
 TrPnt está el indexado hacia el primer movimiento de Tree.*/
{
 short u, sq, xside;
 struct leaf far *node;
 unsigned char far *ppos, far *pdir;
 short i,piece,*PL,r7,r6;
 char tipo,signal;
 unsigned char nodero,sv;

//limitación "subjetiva" para la generación de "extramovimientos"
 signal = (ply - Sdepth > 3) ? 1 : 0;
 nodero = NodeCnt;
 xside = otherside[side];
 TrPnt[ply + 1] = TrPnt[ply];
 node = &Tree[TrPnt[ply]];
 r7 = rank7[side];
 r6 = rank6[side];
 PL = PieceList[side];
 //escruta todas las piezas del mismo bando una a una
 for (i = 0; i <= PieceCnt[side]; y++)
  {
   sq = PL[i];
   piece = board[sq];
   tipo = jaqtipo [piece];
   if (sweep[piece])  // se trata de una pieza tipo dama,alfil o torre
    {
     ppos = nextpos+piece*64*64+sq*64;
     pdir = nextdir+piece*64*64+sq*64;
     u = ppos[sq];
     do
      {
       if (color[u] == neutral) //la casilla está vacia, no existe captura
       {
        if ((jaqtab [u] == nodero) && (jaqpub [u] & tipo))
        { switch (piece) {
     case 3 : sv = jaqalf [u]; break;
          case 4 : sv = jaqtorr [u]; break;
          case 5 : sv = jaqdam [u]; break;
          }

         if (!signal || sv > 127)
           Link (sq,u,0,sv);      //introducción (linkado) de la jugada en Tree
       }
       u = ppos[u];
      }
      else
       {
```

Tabla 2. (continuación

```
if (color[u] == xside) //si se trata de una pieza contraria se puede capturar
{
 if ((jaqtab [u] == nodero) && (jaqpub [u] & tipo))
 { switch (piece) {
  case 3 : sv = jaqalf [u]; break;
  case 4 : sv = jaqtorr [u]; break;
  case 5 : sv = jaqdam [u]; break;
  }

  Link (sq, u, capture,
       value[board[u]] + svalue[u] - piece + sv);
 }
 else
 Link (sq, u, capture,
      value[board[u]] + svalue[u] - piece);
}
u = pdir[u];
}
} while (u != sq);
}
else

//resto de ejemplo para peones, caballos y rey
............

}
}
```

Diagrama 1, mueven negras.

Si el programa hiciera un análisis sin extensores a obtendríamos:
PN 1:
1. Dg7xg2; **Ev: +310**

PN2:
1. Dg7xg2; 2. Ae2xh5, Cf6xh5; **Ev: +310**
PN3:
1. Cc6xd4; 2. Dd2x24; Dg7xg2; **Ev: +100**
PN4:
1. Cc6xd4; 2. Dd2x24; Dg7xg2; 3.Th1g1 **Ev: +100**

Tablero de referencia de movimientos

a8	b8	c8	d8	e8	f8	g8	h8
a7	b7	c7	d7	e7	f7	g7	h7
a6	b6	c6	d6	e6	f6	g6	h6
a5	b5	c5	d5	e5	f5	g5	h5
a4	b4	c4	d4	e4	f4	g4	h4
a3	b3	c3	d3	e3	f3	g3	h3
a2	b2	c2	d2	e2	f2	g2	h2
a1	b1	c1	d1	e1	f1	g1	h1

PN5:
1. Ah5xe2; 2. Dd2xe2. **Ev: -25**

Las variantes principales obtenidas nos muestran los errores que se pueden llegar a cometer por efecto del corte de la PN sobre una idea que continua más allá. El quit de esta posición reside en el hecho de que la dama negra no puede tomar en g2 porque una de las dos torres blancas la clava de tal modo que al intentar escapar de la tenaza de la torre blanca su rey queda en jaque. Bajo PN 1 y 2, el programa toma en g2 dándose la ventaja de una pieza (Ev : +310). En PN 3 se da cuenta del error la secuencia Dxg2; Th1g1 surte su efecto el negro se ve forzado a mover y ve que no puede apartar la dama cuando en ply 4 (en la secuencia de búsqueda de capturas) el banco hace Txg2+ ganando. El ordenador encuentra otra combinación válida y piensa que gana el de peón d4 siguendo la variante descrita para PN 3.

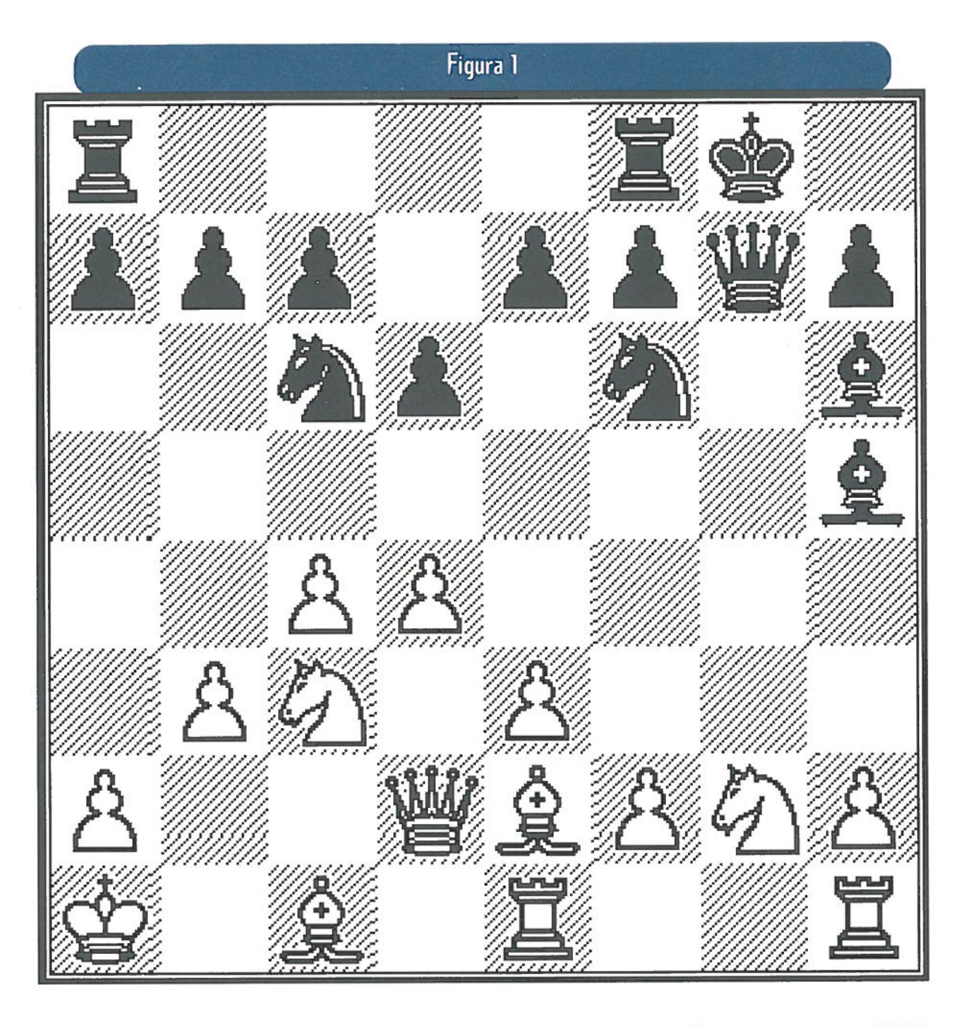
Un humano no le encontraría sentido al sacrificio del caballo y menos si se le dijera que es para capturar en g2 con la dama, el efecto horizonte (ceguera por límite de PN) se debe a que de nuevo el ordenador no ve el peligro de Th1g1 debido a que ha perdido dos jugadas de su PN calculando la falsa Cc6xd4 y por ello no ve las consecuencias posteriores. En su lugar realiza un doble error.

No es hasta PN 5 (pura, sin extensores) cuando se da cuenta de todo y cambia de parecer eligiendo Ah5xe2 con ligera desventaja como línea principal.

Si seleccionamos un sistema con extensores el programa identificaría el error de PN 3 en el nivel 4 y no en el 5, siempre que estuviera activado el extensor de recapturas (se produce una recaptura sobre d4). El extensor por piezas colgadas no se activa cuando está amenazada la dama porque la dama posee huidas como la casilla h3, si no hubiera sido este el caso entonces el error de PN 3 se habría identificado por tener dos extensiones la variante.

Como hemos visto en este ejemplo los extensores han ayudado a salvar una situación un ply por debajo de lo normal,

Figura 1



lo cual en una competición puede ser una ventaja ganadora. (Ver figura 2).

Diagrama 2, Mueven negras

En esta posición las negras tienen un ataque decisivo sobre g3, la disposición de los dos alfiles y la dama del bando negro propician un sacrificio de alfil destruyendo la barrera de peones que cubre al rey blanco. La secuencia sería:

1. Ad6xg3; 2 Pf2xg3, Dxg3 3. Ag5xf6, Ad7xh3 creando serias amenazas

Si ésta fuera la secuencia "real" (se trata de la "ideal") la jugada ganadora sería encontrada forzosamente en PN 6 (Ad7xh3 es la 5° jugada luego se hace forzoso un movimiento blanco para hacerse consciente de la inclinación material de la balanza).

Los extensores han ayudado a salvar una situación un ply por debajo de lo normal

Pero ocurre que el "entre-movimiento" 3. Ag5xf6 se realiza en la 2. jugada con lo que se alarga la combinación para el ordenador puesto que el blanco no retoma en g3 como sería lo deseable (para que la dama negra retome en 3. Dxg3 obteniendo buena posición). De este modo la solución no se vería hasta PN 7 u 8 dependiendo de la cantidad de "entre-movimientos" que generase el ordenador.

Gracias a los extensores la solución puede ser vista en PN 5 ó 6 dependiendo de las opciones escogidas. Una diferencia de 2 pliegues a niveles altos (por encima de PN 6 hasta 9) puede suponer con toda seguridad más de un minuto de tiempo. Y un análisis a PN 5 puede suponer 5-10 segundos.

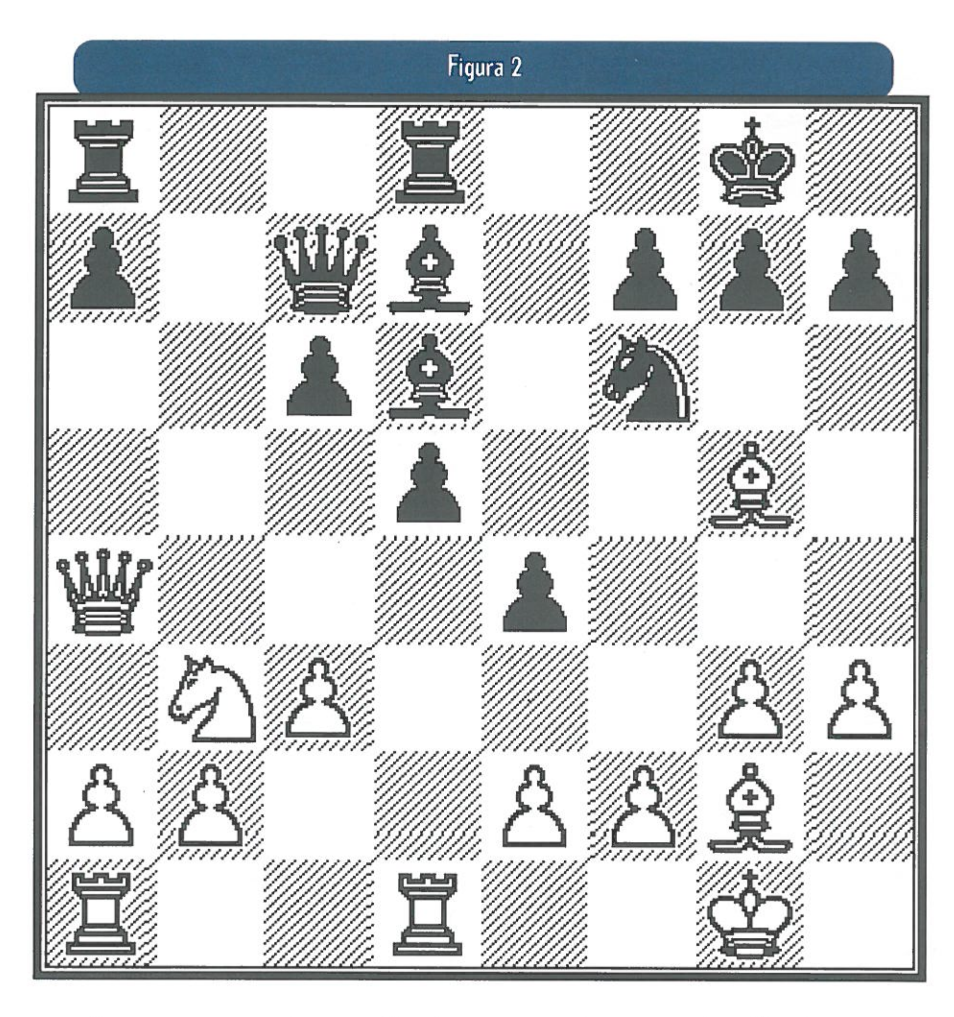
Existen sistemas basados en otras filosofías más "humanas" que resuelven problemas a PN fantásticas

Movimiento Selectivo

Como hemos visto hasta ahora los extensores han supuesto un gran avance en la implementación de la fuerza de juego de los programas, sin embargo existen sistemas basados en otras filosofías más "humanas" que resuelven problemas a PN fantásticas. Por ejemplo en el diagrama 1 la jugada errónea Dg7xg2 junto con la posterior (ahora identificada como entre-movimiento) Cc6xd4, es identificado en PN 1, y el problema del diagrama 2 en PN 2. Recordemos que hasta PN 5 no se veía claramente la solución.

Esto lo consiguen los programas basados en el movimiento selectivo (MS a partir de ahora), su filosofía de juego se basa en identificar posiciones críticas y forzar a un bando a mover fuera de PN (osea en la búsqueda de capturas) sólo aquellas jugadas que sean "lógicas" aparte de las capturas obligatorias.

Figura 2



Supongamos que estamos analizando el diagrama 1 a PN 1, cuando nuestro EX (programa basado únicamente en extensores) da como buena 1. Dg7xg2, el programa MS genera en la búsqueda de capturas la jugada "táctica" Th1g1 (que no es una captura) y ya es consciente que ha clavado la dama, a continuación fuerza un turno equivalente a uno que se encuentre dentro de PN y escoge sólo las jugadas de huida de dama o que cubran a ésta. Finalmente llega a la conclusión de que 1. Dg7xg2 es mala en milésimas de segundo. Otro tanto sucede con el segundo diagrama.

La programación de este tipo de ingenios no es nada fácil, un EX se puede programar en pocos meses y un MS requiere varios años de esfuerzo puesto que la cantidad de jugadas aparentemente lógicas es muy rica en situaciones y matices.

La principal diferencia entre un MS y un EX reside en el hecho de que el primero al ver una situación de peligro o prometedora extiende el ámbito de la "fuerza bruta" y el segundo sólo realiza las que considera necesarias, produciendo ahorros considerables de nodos. Un pliegue sólo con movimientos seleccionados puede considerar de tres a cinco movimientos y otro de fuerza bruta de veinte a cincuenta, además hemos de considerar el hecho de que posteriormente el árbol se propaga exponencialmente.

Naturalmente esta magia aparente posee un precio y es que los MS suelen ser más lentos generando jugadas que los EX puesto que necesitan mover un volumen de información muy superior, información que por otra parte se suele aprovechar para realizar evaluaciones más exactas lo cual es muy beneficioso en los finales donde el conocimiento posicional suele ser más provechoso que el puramente táctico.

El GNU y sobre todo el CRAFTY son claros exponentes de los EX

Hasta ahora podíamos decir que un programa era superior a otro si en el mismo periodo de tiempo alcanzaba una PN superior, pero a la vista está que los MS rompen este principio. Si tenemos la oportunidad de comprobar los análisis de ambos tipos de programas acerca de una posición veremos que el EX rápidamente alcanza PN 6 ó 7 y luego avanza lentamente hasta PN 11 (en máquinas Pentium 200Mhz) sin embargo los MS se quedan habitualmente clavados en PN 6 a 7 alcanzando sólo a veces PN 9 (hablamos de situaciones de medio juego, en los finales esto se equilibra). Un campo en el que los EX poseen ventaja es en el de los problemas "ilógicos" donde las jugadas sorpresivas son las ganadoras, si realizamos estadísticas de tiempos podremos ver casos como EX resuelve en 10 segundos y MS en 240 s.

Los MS suelen ser más lentos generando jugadas que los EX

Por los resultados obtenidos en las competiciones realizadas en los últimos quince años los MS se han mostrado superiores a los EX, sin embargo si consideramos sólo los últimos 5 años la situación es de ligera ventaja para los primeros. Otro hecho a tener en cuenta es el tema de los programas híbridos que son perfectamente válidos y extremadamente peligrosos. Uno de los máximos exponentes y pioneros de la escuela MS es Richard Lang autor del antiguo Psion-Chess, algunos modelos Mephisto y actualmente del programa Genius, le siguieron en este campo Marti Hirsh (Mchess) y Marck Uniacke (Hiarcs) entre otros.

Evidentemente el GNU y el CRAFTY (ante todo este último) son claros exponentes de los EX.

OPINIÓN

Atacando al efecto 2000

J. J. Embid

En este artículo intentaremos dar soluciones al Efecto 2000 y minimizar e incluso evitar los daños producidos por este problema informático actual.

Por empezar por alguno de los múltiples problemas que se producirán, comentaremos que según el chip del BIOS instalado, en algunos equipos el reloj del ordenador, cuando llegue a su límite máximo, se podría inicializar al primer valor o fecha base del sistema. Se podría dar el caso de que cuando el reloj del sistema alcance el 31/12/1999 no cambie a 1/1/2000. Podría saltar a 1.900 o a cualquier otra fecha. Se tiene información de que en los equipos Intel, el MS-DOS, cuando el reloj llegue a su límite máximo, saltará a 1980 o 1984. Por otro lado y confirmado por IBM -tanto en su Web como en sus comunicados- necesita adaptar sus mainframes basados en el procesador S390, a la "compatibilidad del año 2.000".

En cuanto al almacenamiento de datos, podría haber problemas en relación a la fecha límite de almacenamiento en cinta (fecha de retención). Por ejemplo, si queremos que se guarde indefinidamente una cinta le ponemos fecha de retención 99365, por lo que el 31/12/99 se borraría. Podría fallar la copia el 1/1/2000, cuando la etiqueta llevase la indicación 00001. Se prevé que en los ordenadores más antiguos se necesitará ejecutar un comando "DATE" cada vez que se inicie el sistema.

El 2000 y el Hardware del PC

Aparte del software tocado por el efecto 2.000, el hardware del PC también puede verse afectado y los resultados serían desastrosos si no actuamos, sobretodo en ciertos equipos. Los esfuerzos por acabar con el problema en el software del PC pueden ser inútiles si no se toma el punto de vista desde el hardware.

Ya muchas compañías fabricantes aplican la solución al problema en sus nuevos equipos y, realizan una actualización de la BIOS - Sistema de Entrada/Salida Básico - que colocan en los nuevos sistemas.

La mayoría de los PCs tienen un temporizador hardware, que está alimentado con una batería, llamado reloj de tiempo real RTC (*Real Time Clock*) para mantener la fecha y hora del sistema aunque éste se apague. Recordar que la mayoría de los equipos PCs AT y los basados en 286 siempre arrancaban con la fecha 1/1/1980 al no tener su reloj una batería. El RTC fue diseñado para almacenar únicamente los últimos dos dígitos de la fecha. Para superar esta deficiencia, se sumaba un byte en memoria CMOS para almacenar los otros dos dígitos de la fecha. Cuando el sistema es reinicializado, la memoria ROM combina el RTC y el byte de la CMOS para crear el año de cuatro dígitos, el cual es pasado al sistema. En los

La llegada del nuevo milenio, llamado Efecto 2.000, también Y2K en abreviatura inglesa, plantea importantes problemas para la mayoría de los sistemas de información, empezando en el nivel de equipos hardware, software de base, hasta llegar a las aplicaciones finales y los datos.

números atrasados

SÓLO PROGRAMADORES

completa ya tu colección





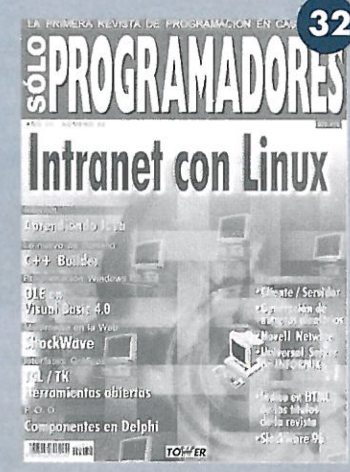








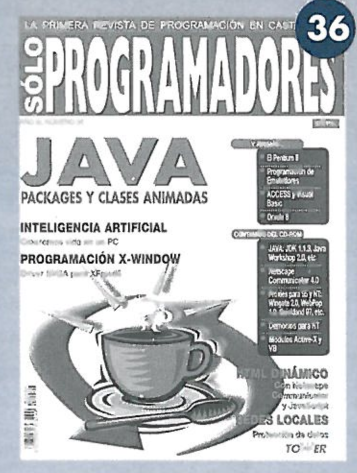












PCs, la actualización del byte CMOS está inhabilitada lo que conduce a la interpretación incorrecta del año 2000. La RTC se actualiza por si misma desde "99" a "00", pero la CMOS se mantiene constante en "19", lo que en teoría causa que el 2000 se convierte en el 1900.

En realidad, dependiendo de las versiones del chip BIOS y del sistema operativo utilizado, la fecha del sistema probablemente será interpretada como "1980" o "1984". Por lo cual los equipos producirán fechas incorrectas después del 31 de diciembre de 1.999.

Normalmente las BIOS fabricadas a finales del 95 y ensambladas en el primer cuatrimestre del 96 - indicadas por 1Q96 - ya están adaptadas al año 2.000.

Los fabricantes y emsambladores proveen una actualización de la BIOS para corregir los problemas del sistema o realizan ampliaciones al sistema. Las actualizaciones son distribuidas generalmente vía Internet -Web o Ftp -.

Otro método para corregir el problema es cambiar la fecha manualmente, mediante un comando del sistema operativo. Utilizando el comando DATE - entorno MSDOS - o la función fecha y hora del panel de control - entornos Microsoft Windows 95 y Windows NT -, podemos cambiar el byte de la CMOS de "19" a "20". Entramos en el icono Mi PC y Panel de Control y modificamos fecha.

Lo malo de este método de actualización es que se necesita la interacción del usuario, aunque mínima, e idealmente debería ejecutarse en el instante en que se inicie el año 2000, permitiendo una pequeña oportunidad de actualización.

Esta solución que algunos colegas puristas sugieren, opino que no es recomendable y debería utilizarse sólo en el caso en que la actualización de la BIOS fuese inviable.

Fabricantes que utilizan la BIOS adaptada al año 2000

MARCA	BIOS
Acer	3Q95
Apple	No afectado
AST	2Q96
Compaq	2Q96
Dell	2Q96
H.P.	1Q96
I.B.M.	1Q96
Olivetti	1Q96
Toshiba	2Q96

Soluciones en el Software

Y nos referimos en este apartado a todo tipo de programas de software base, de comunicaciones, de bases de datos, de ofimática, de herramientas, utilidades, etc.

Algunos de los problemas que se pueden plantear pueden ser derivados del primer apartado, el hardware no preparado: si el sistema nos da una fecha errónea, significará que todo el software del equipo tendrá la fecha equivocada.

A este respecto algunas empresas como, por ejemplo, IBM y Microsoft nos han informado sobre una serie de productos que sí soportan el año 2.000, pero es necesario tener información de la totalidad de los productos instalados en nuestros equipos particulares y del trabajo que realizan.

Normalmente las versiones de IBM y Lotus anteriores a 1.997 o con más de dos años de antigüedad no valdrán. De este modo, por ejemplo el Lotus Notes en versiones anteriores a la 4 nos darán problemas. Igualmente Microsoft en su software realizado antes del 1.997 nos avisa de una serie de limitaciones que tendremos que tener en cuenta para evitar sorpresas.

Soluciones en las Aplicaciones finales

Atendiendo al tipo de T.I de gestión, técnica, gráfica, etc., varía el enfoque. Así en la Informática de gestión - sistemas corporativos, sistemas centralizados con desarrollo distribuido, sistemas de información locales y de usuario final -, tendremos los problemas derivados del hardware no listo para el 2.000. Si el sistema, ya sea por la BIOS, ya sea por el tipo de reloj central, nos da una fecha errónea, significará que todos los programas tratarán una fecha errónea. Aunque las aplicaciones estén bien diseñadas y contemplen el año de las fechas con cuatro dígitos, estas producirán errores.

Así en los grandes sistemas, tipo IBM, con un ordenador mainframe 3090 esto supondrá la conversión de los programas codificados en COBOL I, la mayoría, y COBOL II a COBOL for MVS&VM con el fin de que sigan funcionando. Aparte, hay que tener en cuenta los programas con PLI, Fortran y las conexiones con rutinas con lenguaje emsamblador de cada sistema.

Otro tipo de problemas surgirán por el software inadecuado. Nuevas subidas de versión en el hardware y/o software base implicaría subidas de nivel en las aplicaciones finales así como incompatibilidades entre versiones.

Un problema típico de los dígitos usados en el campo año es el siguiente: en los años 60, 70 y 80 los programadores diseñaron las aplicaciones utilizando el formato estándar de fechas con el año de dos dígitos para ahorrar espacio de almacenamiento, olvidando la llegada del año 00. Esto va a provocar problemas en cálculos aritméticos y comparaciones, por ejemplo: cálculos efectuados teniendo el valor 00 como denominador, cálculo de años por diferencia, al ordenar el año 2000 no estaría después del 1999, etc. Habrá que rediseñar los sistemas pasando de dos a cuatro dígitos los campos

numéricos de almacenamiento y en salidas de impresora o visualización incluyendo el punto de los miles, en muchos casos.

Además de la I. de Gestión, habrá Efecto 2.000 en el resto de las áreas. Así en la I. Gráfica tendremos problemas en los programas de Ingenierías de Diseño y Fabricación/CAD-CAM, gráficos 3D. Es necesario revisar las aplicaciones y productos enfocados a la informática gráfica, como los productos de AutoCAD/AutoDesk. Incluso existirán problemas en los programas para realizar gráficos para las Webs de Internet, ya sea usando programas editores tipo HotDog o HotMetal. El peligro lo representan las versiones anteriores al 97.

En la I. Técnica, lo primero es generar también un inventario por cada equipo contemplando todos los sistemas o subsistemas donde el impacto del año 2.000 pueda ponerse de manifiesto. Se podría clasificar este inventario en distintos apartados en función de sus características - programas de simulación de situaciones, comportamiento de fluidos, etc. Programas de estructuras en FORTRAN y NASTRAN con cálculos de fechas y salidas por impresora, pantalla o terminales de fabricas, máquinas o procesos industriales. Así en los programas usados en Ingeniería de Planta hay que estudiar los sistemas que puedan tener algún control por fecha en las instalaciones . Los programas de accesos, control de gasto eléctrico, telefónico, aire acondicionado, mantenimiento de máquinas, etc. Por ejemplo el mal funcionamiento de los sistemas de refrigeración y seguridad de la sala de ordenadores el día 1/1/2000 supondría un grave problema en las instalaciones informáticas.

Problema en el Nivel de los datos

La conversión de fechas de dos a cuatro dígitos en los programas supondrá conversión de datos actuales así como de datos históricos. La mayoría de estos últimos en bancos, industrias grandes, universidades, etc. se encuentran almacenados en cinta magnética. La recuperación de datos históricos se hace necesario por requerimientos de auditorías, inspecciones fiscales o consultas de datos históricos y cuando queremos buscar algo que no esté en los discos duros. Igual sucede en los equipos PCs pequeños. Normalmente tendremos varios tipos de fecha y no siempre la misma. Así lo normal es el formato DDMMAA, o sea dos dígitos para día, dos para mes y dos para año. Otros diseñadores previsores habrán diseñado como AAAAMMDD con 4 dígitos para año, o bien DDMMAAAA.

Otros programas del área anglosajona trabajarán con el formato tipo AAMMDD, que luego al visualizarlo hay que dar la vuelta, en el area hispana. Aparte está el tema de los separadores. Si en las multinacionales con programas en inglés es normal ver 97/02/28 que para el área española y sudamericana es 28-2-97 por ejemplo.

Otras veces el dato está almacenado sin contar con el año, porque este daño es genérico y se indica en el nombre del fichero y así es PEDIDOS96 o ALMACEN95 y los datos de fecha van en formato DDMM o MMDD. Todo eso hay que analizarlo, ordenarlo y organizarlo. Por eso preveían todas las grandes auditorías unos importantes volúmenes de trabajo que no podrían hacer frente los centros de cálculo de muchas empresas. Se comenta que un porcentaje de empresas desaparecerán entre pleitos informáticos y colapsos de sus sistemas informáticos. O aprovechándose del conocido Efecto 2000 cerrarán sus puertas.

A todo lo anterior se une el problema del año bisiesto - el año 2000 no es bisiesto aunque sea múltiplo de 4 - y la introducción del EURO. Por convención el primer año de todos los siglos nunca es un año bisiesto.

En el tema de las redes de ordenadores, se podría dar el caso de que un programa de un nodo en la red rechace el 29 de febrero del 2.000 con lo que toda la red quedaría inservible durante ese día y las transacciones podrían no armonizarse hasta pasado bastante tiempo. Habría que revisar en todos los procesadores de texto, hojas de cálculo, bases de datos, agendas y otros paquetes de uso cotidiano sí el día siguiente al 28-2-2000 es el 1-3-2000. En algunos equipos, el día siguiente al 28 de febrero de 1900 es también el 29 de febrero en lugar del 1 de marzo.

El problema del EURO - que todavía no sabemos nada más que en muchos sitios tendrá que ser efectivo el 1 de enero del 99 - no conocemos su nomenclatura o acrónimo. ¿Será EUR, será una e rara tipo Epsilon griega como he leído en algunas revistas?

Este tema afectará sólo a las aplicaciones finales, suponiendo modificaciones de programas (aumento de dígitos en los decimales de los campos de importes, eliminación de redondeos), modificación de soportes de almacenamiento, análisis de nuevos procesos, revisión de los procesos actuales, depuración de procesos obsoletos y adaptación de los sistemas de la compañía a la nueva normativa. Ahora despreciamos los decimales de peseta que no valen nada, pero un céntimo de euro vale unas cuantas pesetas. Además tenemos que tener en cuenta el aspecto del separador, coma en nuestro caso, para separar los enteros de los decimales.

¿Está seguro de que en enero del año 2.000 podrá usted trabajar con completa normalidad?. Lo dudamos. Cosa buena, habrá trabajo para bastantes analistas y programadores.

WEB de interés:

IBM http://www.software.ibm.com/year2000/resource.html
Digital http://www.digital.com
Gartner http://www.gartner.com
Advantage http://www.advantage.com
Year http://www.year2000.comIndice a otros-recomendable

Españoles :

Indra http://www.indra-si.es
Estrella http://www.ctv.es/USERS/estrella/2000.htm recomendable.

El lenguaje unificado para modelado

Marta Huertas

Con este artículo pretendemos dar cabida en Sólo Programadores a una parte del proceso de desarrollo de software que quizá esté un poco olvidada en la prensa especializada, el análisis y diseño de software, pero no por ello es de menor importancia. Conozcamos qué es UML.

Desde que con Simula 67 se introdujeran los conceptos de clase y herencia allá por los años setenta, los lenguajes de programación OO han experimentado una rápida e intensa evolución para acercarse a la realidad que la empresa les reclamaba. La extendida utilización de lenguajes como el C++ o el Delphi, y la creación de otros nuevos que incorporan desde el principio de su aparición características OO, como es Java, parecen indicar que en la ingeniería del software ha tocado fondo en lo que a novedades metodológicas se refiere. Las famosas bases de datos relacionales llevan algún tiempo incorporando las siglas OO a las suyas propias, así, SGBDROO es ahora un acrónimo conocido que tiene en el mercado algunos productos que están dando mucho que hablar, como los nuevos servidores Universal Server de Informix, que aplican la OO mediante los llamados *datablades*, módulos de componentes software que intentan abarcar todas las necesidades en cuanto a tipos de datos complejos se refiere. Esta misma tendencia se viene apreciando desde hace mucho tiempo en otros campos de la informática, ¿es que no hay nada mejor que la OO para el desarrollo de sistemas informáticos?

Mientras las cosas continúen así, bien harán los profesionales del sector manteniéndose informados respecto a lo que en OO se cuece. Y parece que en los últimos dos años, aproximadamente, algo muy importante está en el horno.

Existen cientos de herramientas CASE que utilizan las metodologías OO como base para el diseño de aplicaciones

Por todos es conocido que existen muchas metodologías OO que se utilizan en la empresa y que paralelamente aparecen cientos de herramientas CASE que las usan como base para el diseño de aplicaciones de todo tipo. Algunas de las metodologías más importantes son la Booch '93, la OMT (*Object Modeling Technique*) con sus actualizaciones (como la OMT-2) y la OOSE (*Object Oriented Software Engineer*). Y precisamente los principales autores de estas metodologías y notaciones gráficas – en la creación de las metodologías suelen intervenir varias personas – llevan algunos años trabajando para conseguir que sus principios se unifiquen y establecer un estándar sobre el cual trabajar. Las razones para esta unificación son varias pero una de las principales es el clima de confusión que se estaba generando: los diseñadores cogían a su antojo las características que necesitaban, buscando suplir las carencias de los métodos que utilizaban. Así, la utilización de varios métodos a la vez para aprovechar las ventajas de unos y otros era una práctica generalizada, lo

cual constituye una mecánica de trabajo, cuando menos, peligrosa, al conducir a una situación en la que los propios desarrolladores no son capaces de discernir qué características procedían de qué métodos.

Los orígenes

A partir del año 1994, Grady Booch (como su nombre indica, precursor de Booch '93) y Jim Rumbaugh (creador, entre otros, de OMT) se unen en una empresa común (de objetivos y de negocio), y dentro de la *Rational Software Corporation* comienzan a unificar sus dos métodos. Un año después, en octubre de 1995, aparece UML (*Unified Modeling Language*) 0.8, la que se considera primera versión del UML. A finales de ese mismo año, Ivar Jacobson (creador de OOSE) se añade al grupo.

Los principales creadores de UML son Grady Booch, Jim Rumbaugh e Ivar Jacobson

Como objetivos principales de la consecución de un nuevo método que aunara los mejores aspectos de sus predecesores, sus protagonistas se marcaron los siguientes:

- El método debía ser capaz de modelar no sólo sistemas software sino otro tipo de sistemas reales de la empresa, siempre utilizando los conceptos de la orientación a objetos.
- Crear un lenguaje para modelado utilizable a la vez por máquinas y por personas.
- Establecer un acoplamiento explícito de los conceptos y los *artefactos* ejecutables.
- Manejar los problemas típicos de los sistemas complejos de misión crítica.

Las versiones siguientes estuvieron pronto a disposición de quien las quisiera (de hecho, toda la especificación del UML se puede encontrar en Internet, consúltese bibliografía para obtener información) y las versiones 0.9 y 0.91 sirvieron como base, al ser expuestas a toda la comunidad de desarrolladores, para continuar trabajando sobre el UML.

Durante 1996, varias compañías se unieron a los esfuerzos de creación de la definición del UML. De esta forma se crea un consorcio de empresas que con su experiencia contribuyen a la creación del UML 1.0. La importancia de estas empresas puede ayudarnos intuir la relevancia que puede llegar a adquirir el UML: Intellicorp, IBM Oracle, Rational Software, HP, Digital Equipment Corp., i-Logix, Icon Computing, MCI Systemhouse y Unisys forman el consorcio para el desarrollo del UML. Con su participación se ha pretendido obtener otras perspectivas al formar parte activa del proyecto. Así se incluyen en UML perspectivas sugeridas por el OMG, tecnologías RM-ODP, para realizar modelado de negocio, semántica de máquina de estado (mediante la incorporación de diagramas de estado), tipos, interfaces, colaboraciones, refinamiento, distribución y un metamodelo para definir otros modelos.

EL resultado fue la versión final 1.0 (por el momento), que se encuentra en fase de ser aceptada como estándar por el OMG (*Object Management Group*). Requisito básico para obtener esta estandarización es que UML no sea propiedad de ninguna empresa (*Rational Software Corporation* comercializa una herramienta CASE de la que hablaremos después pero no posee derechos sobre UML) y los creadores admiten sugerencias por parte de los desarrolladores y las personas que lo utilicen para conseguir que el lenguaje se convierta en un estándar abierto y utilizable en el desarrollo de todo tipo de aplicaciones sin restringir ninguna en concreto. La creación del UML pretende conseguir además dos objetivos claramente definidos: por un lado, que los lenguajes que se aplican siguiendo los métodos más utilizados sigan evolucionando en conjunto y no por separado, como estaba ocurriendo hasta ahora, y se concluyan de esta forma las diferencias entre ellos; y por otro lado, y lo que puede en determinadas situaciones ser de extrema importancia, unificar las perspectivas entre diferentes tipos de sistemas (no sólo software, sino también de ámbito de negocio), al aclarar las fases de desarrollo, los requerimientos del análisis, el diseño, la implementación y los conceptos internos de la OO.

UML 1.0

La versión de UML final hasta el momento es la 1.0 que constituye una versión estable y utilizable, cuya especificación completa se encuentra a disposición del público en internet; este artículo está basado en esos mismos documentos que *Rational Software Corporation* mantiene para facilitar el estudio del lenguaje. Se espera que el OMG decida sobre su estandarización en lo que queda de este año. Mientras esto sucede, existen diversos grupos en Internet donde se discute sobre el lenguaje (por ejemplo el grupo de noticias comp.object). Si desea enviar comentarios sobre el UML al grupo de desarrollo de *Rational Software* puede hacerlo a través de la dirección de correo electrónico: uml_feedback@rational.com.

El UML es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de software OO, así lo definen sus creadores. Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software. Definida la palabra que quizá causara mayor confusión en la definición de UML, el resto no deja sitio para la ambigüedad, el UML se quiere convertir en un lenguaje estándar con el que sea posible modelar todos los componentes del proceso de desarrollo de aplicaciones. Sin embargo, hay que tener en cuenta un as-

pecto importante del modelo: no pretende definir un proceso estándar de desarrollo, sino únicamente un lenguaje de modelado. Otros métodos de modelado como OMT o Booch sí definen procesos concretos, la razón que argumentan sus creadores para esta omisión dentro del UML es que los procesos de desarrollo son diferentes según los distintos dominios de trabajo; no puede ser el mismo el proceso para crear una aplicación en tiempo real, que el proceso de desarrollo de una aplicación orientada a gestión, por poner un ejemplo. Las diferencias son muy marcadas y afectan a todas las fases del proceso. El método del UML recomienda utilizar los procesos que otras metodologías tienen definidos.

El UML es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de software OO

Cuando se fusionan elementos diferentes dentro de cualquier campo de la ciencia, lo que se pretende resulta evidente: eliminar aquellos componentes que sean de poca utilidad que no se utilicen en la práctica, añadir componentes conocidos de otros elementos que supongan una mejora efectiva e inventar soluciones para situaciones que ninguno de los elementos anteriores puedan resolver por sí mismos. Todo ello intentando mantener la simplicidad precisa para que, en el caso que nos ocupa, el método de trabajo sea utilizable. Asimismo, la migración desde otras metodologías al UML no debe ser traumática, es decir, no debe exigir cambios importantes, lo cual se consigue debido a que el lenguaje no supone un cambio radical con respecto a los métodos anteriores, sino una evolución de los mismos. Se pretende que los desarrolladores acostumbrados a trabajar con otras metodologías puedan adaptar su forma de trabajar sin apenas esfuerzo.

Pero un estándar no se convierte en tal únicamente porque sus creadores decidan que lo sea, el proceso lleva tiempo y sobre todo conlleva el que sea utilizado por un número importante de personas en sus proyectos de desarrollo. Deberán ser los propios usuarios de metodologías OO los que decidan si este método es el adecuado con su experiencia y sus conocimientos. Por lo pronto, y en lo que a nuestro país se refiere, el UML ha aparecido ya como una exigencia de conocimiento (suponemos y esperamos que con las adecuadas reservas; es curioso observar cómo en algunos anuncios piden 5 años de experiencia en desarrollo en Java) en algunas ofertas de trabajo de las que se publican en los medios de comunicación. Esto puede llevarnos a pensar que esta metodología dará mucho que hablar próximamente.

Modelado de objetos

En la especificación del UML podemos comprobar que una de las partes que lo componen es un metamodelo formal pero, ¿qué es un metamodelo? Un metamodelo es un modelo que define el lenguaje para expresar otros modelos. Un modelo en OO es *una abstracción cerrada semánticamente de un sistema* y un sistema *es una colección de unidades conectadas que son organizadas para realizar un propósito específico*. Un sistema puede ser descrito por uno o más modelos, posiblemente desde distintos puntos de vista.

A la vista de tales definiciones podríamos deducir que una parte de UML define una abstracción con significado de un lenguaje para expresar otros modelos (es decir, otras abstracciones de un sistema, o conjunto de unidades conectadas que se organizan para conseguir un propósito). Lo que en principio puede parecer complicado no lo es tanto si pensamos que uno de los objetivos del UML es llegar a convertirse en una manera de definir modelos, no sólo establecer una forma de modelo, de esta forma simplemente estamos diciendo que UML, además, define un lenguaje con el que podemos abstraer cualquier tipo de modelo.

El UML es una técnica de modelado de objetos y como tal supone una abstracción de un sistema para llegar a construirlo en términos concretos. El modelado no es más que la construcción de un modelo a partir de una especificación. Un modelo es una abstracción de algo, que se elabora para comprender ese algo antes de construirlo. El modelo omite detalles que no resultan esenciales para la comprensión del original y por tanto facilita dicha comprensión.

El modelado permite abstraer la realidad para eliminar los detalles innecesarios en el proceso de entendimiento de la misma

Los modelos se utilizan en muchas actividades de la vida humana: antes de construir una casa el arquitecto utiliza un plano, los músicos representan la música en forma de notas musicales, los artistas pintan sobre el lienzo con carboncillo antes de empezar a utilizar los óleos,... Unos y otros abstraen una realidad compleja sobre unos bocetos, modelos al fin y al cabo. En esta característica se fundamentan todas las técnicas de modelado; la OMT, por ejemplo, intenta abstraer la realidad utilizando tres clases de modelos OO: el modelo de objetos, que describe la estructura estática; el modelo dinámico, con el que describe las relaciones temporales entre objetos; y el modelo funcional que describe las relaciones funcionales entre valores. Mediante estas tres fases de construcción de un modelo, se consigue una abstracción de la realidad que tiene en sí misma información sobre las principales características de ésta.

Los modelos, además, al no ser una representación que incluya todos los detalles de los originales, permiten probar más fácilmente los sistemas que modelan y determinar los errores. Según se indica en *Metodología OMT* (Rumbaugh et al), los modelos permiten una mejor comunicación con el cliente por distintas razones: es posible enseñar al cliente una aproximación de lo que será el producto final, proporcionan una primera aproximación al problema que permite visualizar cómo quedará el resultado y reducen la complejidad del original en subconjuntos que son fácilmente tratables por separado. Se consigue un modelo completo de la realidad cuando el modelo captura los aspectos importantes del problema y omite el resto. Los lenguajes de programación que estamos acostumbrados a utilizar no son adecuados para realizar modelos completos de sistemas reales porque necesitan una especificación total con detalles que no son importantes para el algoritmo que están implementando. En OMT se modela un sistema desde tres puntos de vista diferentes donde cada uno representa una parte del sistema y su unión lo describe de forma completa. En esta técnica de modelado se utilizó una aproximación al proceso de implementación de software habitual donde se utilizan estructuras de datos (modelo de objetos), las operaciones que se realizan con ellos tienen una secuencia en el tiempo (modelo dinámico) y se realiza una transformación sobre sus valores (modelo funcional).

UML utiliza parte de este planteamiento obteniendo distintos puntos de vista de la realidad que modela mediante los distintos tipos de diagramas que posee. Con la creación del UML se persigue obtener un lenguaje que sea capaz de abstraer cualquier tipo de sistema, sea informático o no, mediante los diagramas, es decir, mediante representaciones gráficas que contienen toda la información relevante del sistema. Un **diagrama** es *una representación gráfica de una colección de elementos del modelo, que habitualmente toma forma de grafo donde los arcos que conectan sus vértices son las relaciones entre los objetos y los vértices se corresponden con otros elementos del modelo*. Los distintos puntos de vista de un sistema real que se quieren representar para obtener el modelo se dibujan de forma que se resaltan los detalles necesarios para entender el sistema.

Artefactos para el Desarrollo de Proyectos

Un artefacto (de la palabra *artifact* utilizada en OO), como decíamos antes, es *una información que es utilizada o producida mediante un proceso de desarrollo de software*. Pueden ser artefactos un modelo, una descripción o un software. Los artefactos de UML se especifican en forma de diagramas, éstos, junto con la documentación sobre el sistema, que forma parte del modelo, constituyen los artefactos principales que el modelador puede observar, aunque el UML además es capaz de proporcionar puntos de vista alternativos.

Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software

La manera en que se modela un sistema para ser construido y la elección de los puntos importantes que se reflejan en el modelo y los que se desechan determinan cómo se solucionará el problema y la dificultad de la implementación posterior. La abstracción, como mencionamos anteriormente, es de vital importancia en este proceso al destacar unos detalles y omitir otros. Se necesita más de un punto de vista para llegar a representar un sistema. UML utiliza los diagramas gráficos para obtener estos distintos puntos de vista de un sistema:

- diagramas de casos de uso
- diagramas de clases
- diagramas de comportamiento o interacción
  - diagrama de estado
  - diagrama de actividad
  - diagrama de secuencia
  - diagrama de colaboración
- diagramas de implementación
  - diagrama de componente
  - diagrama de plataforma

Abarcar todos los aspectos del UML en un único artículo es un esfuerzo vano. Por esta razón, en lo que resta de artículo explicaremos algunos de los conceptos más importantes que encierra esta metodología, pero no nos centraremos en detalles que requerirían un libro para ser entendidos de forma completa. Consúltese la bibliografía para obtener información detallada sobre la especificación completa del UML. Los diagramas de explican en profundidad en la parte del documento de especificación del UML 1.0 denominada Guía de Notación de UML. Todos los elementos del modelo tienen una representación gráfica que no se indica por no ser objetivo de este artículo.

Diagramas de Casos de uso

Un **caso de uso** es *una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema*. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Una relación es una conexión entre elementos del modelo, por ejemplo la relación y la generalización son relaciones.

Los diagramas de casos de uso son similares a los de OOSE. Se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona en respuesta a eventos que se producen en el mismo. En este tipo de diagrama intervienen algunos conceptos nuevos: un **actor** es una entidad externa al sistema que se modela y que puede interactuar con él; un ejemplo de actor podría ser un usuario o cualquier otro sistema. Las relaciones entre casos de uso y actores pueden ser las siguientes:

- Un actor se comunica con un caso de uso.
- Un caso de uso extiende otro caso de uso.
- Un caso de uso usa otro caso de uso.

Diagramas de Clases

Muestra un conjunto de elementos del modelo que son estáticos, como las clases y los tipos, junto con sus contenidos y las relaciones que se establecen entre ellos. En OMT se podría identificar con el modelo de objetos, ya que muestra la estructura estática del modelo. Puede representar la totalidad o parte de la estructura de clases del sistema. Es uno de los diagramas que tiene mayor importancia en la implementación.

Algunos de los elementos que se pueden clasificar como estáticos son los siguientes:

- **Paquete.** Es el mecanismo de que dispone el UML para organizar sus elementos en grupos, representa un grupo de elementos del modelo. Un sistema es un único paquete que contiene el resto del sistema, por lo tanto, un paquete debe poder anidarse, permitiéndose que un paquete contenga otro paquete.

- **Clases.** Una clase representa un conjunto de objetos que tienen una estructura, un comportamiento y unas relaciones con propiedades parecidas. Describe un conjunto de objetos que comparte los mismos atributos, operaciones, métodos, relaciones y significado. En UML una clase es una implementación de un tipo. Los componentes de una clase son:

- **Atributo.** Se corresponde con las propiedades de una clase o un tipo. Se identifica mediante un nombre. Existen atributos simples y complejos.

- **Operación.** También conocido como método, es un servicio proporcionado por la clase que puede ser solicitado por otras clases y que produce un comportamiento en ellas cuando se realiza.

Las clases pueden tener parámetros formales, son las clases denominadas plantillas. Sus atributos y operaciones vendrán definidas según sus parámetros formales. Las plantillas pueden tener especificados los valores reales para los parámetros formales, entonces reciben el nombre de clase parametrizada instanciada. Se puede usar en cualquier lugar en el que podría aparecer su plantilla.

Los diagramas de clases muestran un conjunto de elementos del modelo estáticos, como las clases y los tipos, sus contenidos y las relaciones que se establecen entre ellos

Relacionado con las clases nos encontramos con el término **utilidad**, que se corresponde con una agrupación de variables y procedimientos globales en forma de declaración de clase, también puede definirse como un estereotipo (o nueva clase generada a partir de otra ya existente) de un tipo que agrupa variables globales y procedimientos en una declaración de clase. Los atributos y operaciones que se agrupan en una utilidad se convierten en variables y operaciones globales. Una utilidad no es fundamental para el modelado, pero puede ser conveniente durante la programación.

- **Metaclase:** es una clase cuyas instancias son clases. Sirven como depósitos para mantener las variables de clase y proporcionan operaciones (métodos de clase) para inicializar estas variables. Se utilizan para construir metamodelos (modelos que se utilizan para definir otros modelos).

- **Tipo:** es un descriptor de objetos que tiene un estado abstracto y especificaciones de operaciones pero no su implementación. Un tipo establece una especificación de comportamiento para las clases.

- **Interfaz:** representa el uso de un tipo para describir el comportamiento visible externamente de cualquier elemento del modelo.

- **Relaciones entre clases:** las clases se relacionan entre sí de distintas formas, que marcan los tipos de relaciones existentes:

1. Asociación:
Es una relación que describe un conjunto de vínculos entre clases. Pueden ser binarias o n-arias, según si implican a dos clases o a más. Las relaciones de asociación vienen identificadas por los **roles**, que son los nombres que indican el comportamiento que tienen los tipos o las clases, en el caso del rol de asociación (existen otros tipos de roles según la relación a la que identifiquen). Indican la información más importante de las asociaciones. Es posible indicar el número de instancias de una clase que participan en la relación mediante la llamada *multiplicidad*. Cuando la multiplicidad de un rol es mayor que 1, el conjunto de elementos que

se relacionan puede estar ordenado. Las relaciones de asociación permiten especificar qué objetos van a estar asociados con otro objeto mediante un calificador. El calificador es un atributo o conjunto de atributos de una asociación que determina los valores que indican cuáles son los valores que se asociarán.

- Una asociación se dirige desde una clase a otra (o un objeto a otro), el concepto de **navegabilidad** se refiere al sentido en el que se recorre la asociación.

Existe una forma especial de asociación, la **agregación,** que especifica una relación entre las clases donde el llamado *agregado* indica el todo y el *componente* es una parte del mismo. Un ejemplo que clarifique este tipo de asociación puede darse entre una clase flor y otra pétalo donde la primera es el agregado y la segunda sería el componente que forma parte de la primera.

2. Composición
Es un tipo de agregación (recordemos que a su vez es una forma de asociación) donde la relación de posesión es tan fuerte como para marcar otro tipo de relación. Las clases en UML tienen un tiempo de vida determinado, en las relaciones de composición, el tiempo de vida de la clase que es parte del todo (o agregado) viene determinado por el tiempo de vida de la clase que representa el todo, por tanto es equivalente a un atributo, aunque no lo es porque es una clase y puede funcionar como tal en otros casos.

3. Generalización
Cuando se establece una relación de este tipo entre dos clases, una es una *Superclase* y la otra es una *Subclase.* La subclase comparte la estructura y el comportamiento de la superclase. Puede haber más de una clase que se comporte como subclase.

4. Dependencia
Una relación de dependencia se establece entre clases (u objetos) cuando un cambio en el elemento independiente del modelo puede requerir un cambio en el elemento dependiente.

5. Relación de Refinamiento
Es una relación entre dos elementos donde uno de ellos especifica de forma completa al otro que ya ha sido especificado con cierto detalle.

Diagramas de Interacción o Comportamiento

Muestran las interacciones entre objetos ocurridas en un escenario (parte) del sistema. Hay varios tipos:

1. Diagramas de secuencia
Muestran las interacciones entre un conjunto de objetos, ordenadas según el tiempo en que tienen lugar. En los diagramas de este tipo intervienen objetos, que tienen un significado parecido al de los objetos representados en los diagramas de colaboración, es decir son instancias concretas de una clase que participa en la interacción. El objeto puede existir sólo durante la ejecución de la interacción, se puede crear o puede ser destruido durante la ejecución de la interacción. Un diagrama de secuencia presenta una forma de indicar el período durante el que un objeto está desarrollando una acción directamente o a través de un procedimiento.

Los diagramas de interacción muestran las interacciones que suceden entre objetos

En este tipo de diagramas también intervienen los **mensajes**, que son la forma en que se comunican los objetos: el objeto origen solicita (llama a) una operación del objeto destino. Existen distintos tipos de mensajes según cómo se producen en el tiempo: simples, síncronos, *balking, time-out* y asíncrona.

Los diagramas de secuencia permiten indicar cuál es el momento en el que se envía o se completa un mensaje mediante el **tiempo de transición,** que se especifica en el diagrama.

Este tipo de diagramas son parecidos a los que podemos encontrar en varios métodos de OO con los nombres interacción o traza de mensajes o de eventos.

2. Diagramas de colaboración
Muestra la interacción entre varios objetos y los enlaces que existen entre ellos. Representa las interacciones entre objetos organizadas alrededor de los objetos y sus vinculaciones. A diferencia de un diagrama de secuencias, un diagrama de colaboraciones muestra las relaciones entre los objetos, no la secuencia en el tiempo en que se producen los mensajes. Los diagramas de secuencias y los diagramas de colaboraciones expresan información similar, pero en una forma diferente.

Formando parte de los diagramas de colaboración nos encontramos con objetos, enlaces y mensajes. Un **objeto** es una instancia de una clase que participa en una interacción, existen objetos simples y complejos. Un objeto es activo si posee el *thread* o hilo del control y es capaz de iniciar la actividad de control, mientras que un objeto es pasivo si mantiene datos pero no inicia la actividad.

Un **enlace** es una instancia de una asociación que conecta dos objetos de un diagrama de colaboración. El enlace puede ser reflexivo si conecta a un elemento consigo mismo. La existencia de un enlace entre dos objetos indica que puede existir un intercambio de mensajes entre los objetos conectados.

Los diagramas de interacción indican el **flujo de mensajes** entre elementos del modelo, el flujo de mensajes representa el envío de un mensaje desde un objeto a otro. Un objeto sólo puede enviar un mensaje a otro objeto si entre ellos existe un enlace. Los mensajes que se envían entre objetos puedes ser de distintos tipos, también según cómo se producen

en el tiempo; existen mensajes simples, síncronos, *balking*, *timeout* y asíncronos.

Durante la ejecución de un diagrama de colaboración se crean y destruyen objetos y enlaces.

3. Diagramas de actividad

Son similares a los diagramas de flujo de otras metodologías OO. En realidad se corresponden con un caso especial de los diagramas de estado donde los estados son **estados de acción** (estados con una acción interna y una o más transiciones que suceden al finalizar esta acción, o lo que es lo mismo, un paso en la ejecución de lo que será un procedimiento) y las transiciones vienen provocadas por la finalización de las acciones que tienen lugar en los estados de origen. Siempre van unidos a una clase o a la implementación de un caso de uso o de un método (que tiene el mismo significado que en cualquier otra metodología OO). Los diagramas de actividad se utilizan para mostrar el flujo de operaciones que se desencadenan en un procesamiento interno del sistema.

4. Diagramas de estado

Representan la secuencia de estados por los que un objeto o una interacción entre objetos pasa durante su tiempo de vida en respuesta a estímulos (eventos) recibidos. Representa lo que podemos denominar en conjunto una máquina de estados. Un **estado** en UML es cuando un objeto o una interacción satisface una condición, desarrolla alguna acción o se encuentra esperando un evento.

Cuando un objeto o una interacción pasa de un estado a otro por la ocurrencia de un evento se dice que ha sufrido una **transición**, existen varios tipos de transiciones entre objetos: simples (normales y reflexivas) y complejas. Además una transición puede ser interna si el estado del que parte el objeto o interacción es el mismo que al que llega, no se provoca un cambio de estado y se representan dentro del estado, no de la transición. Como en todas las metodologías OO se envían mensajes, en este caso es la acción la que puede enviar mensajes a uno o varios objetos destino.

Diagramas de Implementación

Se derivan de los diagramas de proceso y módulos de la metodología de Booch, aunque presentan algunas modificaciones. Los diagramas de implementación muestran los aspectos físicos del sistema. Incluyen la estructura del código fuente y la implementación, en tiempo de implementación. Existen dos tipos:

1. Diagrama de componentes

Muestran las dependencias entre los distintos componentes software, incluyendo componentes de código fuente, binario y ejecutable. Un **componente** es un fragmento de código software (sea fuente, binario o ejecutable) que se utiliza para mostrar dependencias en tiempo de compilación o ejecución.

Los diagramas de implementación representan los aspectos físicos del sistema, incluyendo componentes software y hardware

2. Diagrama de plataformas o despliegue

Muestra la configuración de los componentes hardware, los procesos, los elementos de procesamiento en tiempo de ejecución y los objetos que existen en tiempo de ejecución. En este tipo de diagramas intervienen nodos, asociaciones de comunicación, componentes dentro de los nodos y objetos que se encuentran a su vez dentro de los componentes. Un **nodo** es un objeto físico en tiempo de ejecución, es decir una máquina, que se compone habitualmente de, por lo menos, memoria y capacidad de procesamiento, a su vez puede estar formado por otros componentes.

Hasta este punto del artículo hemos revisado (superficialmente) las estructuras principales en las que se fragmenta el modelo que se crea con UML: los diagramas. Mediante la representación de los aspectos que estos diagramas pueden abstraer, se consigue un modelo del sistema que después se utilizará para llegar a una implementación concreta.

Nuevas Características del UML

Los conceptos que se incluyen en UML son en su mayoría extraídos de los métodos de diseño anteriores, UML sólo los ha juntado para que sirvan a un propósito común: llegar desde la abstracción de un problema hasta su solución de forma completa e inteligible por todos los componentes de un equipo de desarrollo. Pero además de los conceptos extraídos de métodos anteriores, se han añadido otros nuevos que vienen a suplir carencias de las antiguas metodologías de modelado. Estos nuevos conceptos son los siguientes:

- Definición de estereotipos: un estereotipo es una nueva clase de elemento de modelado que debe basarse en ciertas clases ya existentes en el metamodelo y constituye un mecanismo de extensión del modelo.
- Responsabilidades
- Mecanismos de extensibilidad: estereotipos, valores etiquetados y restricciones
- Tareas y procesos
- Distribución y concurrencia (para

modelar, por ejemplo ActiveX/DCOM y CORBA)

- Patrones/colaboraciones
- Diagramas de actividad (para reingeniería de proceso de negocio)
- Clara separación de tipo, clase e instancia
- Refinamiento (para manejar relaciones entre niveles de abstracción)
- Interfaces y componentes

El proceso de Desarrollo

Como comentábamos antes, los autores de UML no han definido un proceso concreto que determine las fases de desarrollo de un sistema, las empresas pueden utilizar UML como lenguaje para definir sus propios procesos y lo único que tendrán en común con otras organizaciones que utilicen UML serán los tipos de diagramas. El proceso debe ser determinado según las necesidades propias de la aplicación a desarrollar.

UML no define un proceso estándar para desarrollo

UML es un método independiente del proceso lo cual no significa que sus autores no reconozcan la importancia de que exista un proceso concreto. Determinar correctamente las fases de desarrollo es de vital trascendencia en el desarrollo software, sin embargo, consideran que los procesos de desarrollo deben ser definidos dentro del contexto donde se van a implementar los sistemas. UML puede soportar los procesos que otras metodologías tienen definidos, los procesos de Booch, OMT y OOSE pueden ser utilizados en UML, sin embargo no existe una estandarización al respecto y no es objetivo de UML el conseguirla.

Rational Rose C++

UML es útil para obtener modelos que describan supuestos particulares, pero la definición de un estándar es también de vital importancia en el desarrollo de software debido a que posibilita la generación de herramientas CASE que se fundamentan en dichos estándares y que se utilizan de forma generalizada.

Los creadores de UML tienen muy presente este hecho: todos los métodos OO cuya utilización está extendida en el mercado disponen de herramientas que

facilitan el diseño de los sistemas. A la hora de trabajar sobre la creación de UML sus desarrolladores han tenido siempre en mente cómo debía ser esa herramienta CASE que soportara como anillo al dedo su especificación. Son varios los comentarios al respecto que nos encontramos en los documentos sobre UML, por ejemplo, en UML los componentes genéricos y los concretos se representan utilizando los mismos símbolos geométricos para cada par de elementos (clase-objeto, parámetro-valor,...) y se subraya el nombre cuando el elemento es la instancia del genérico; es el caso del objeto en la pareja clase-objeto. Junto a esta especificación concreta se indica que las herramientas de desarrollo tienen capacidad para sustituir los gráficos para las instancias de estas parejas a elección del usuario, bien podrían utilizar colores distintos, o rellenar el gráfico, por ejemplo. Esta misma libertad la tienen las herramientas CASE de UML en otros muchos casos y se hace hincapié de la misma forma en la propia especificación del UML 1.0.

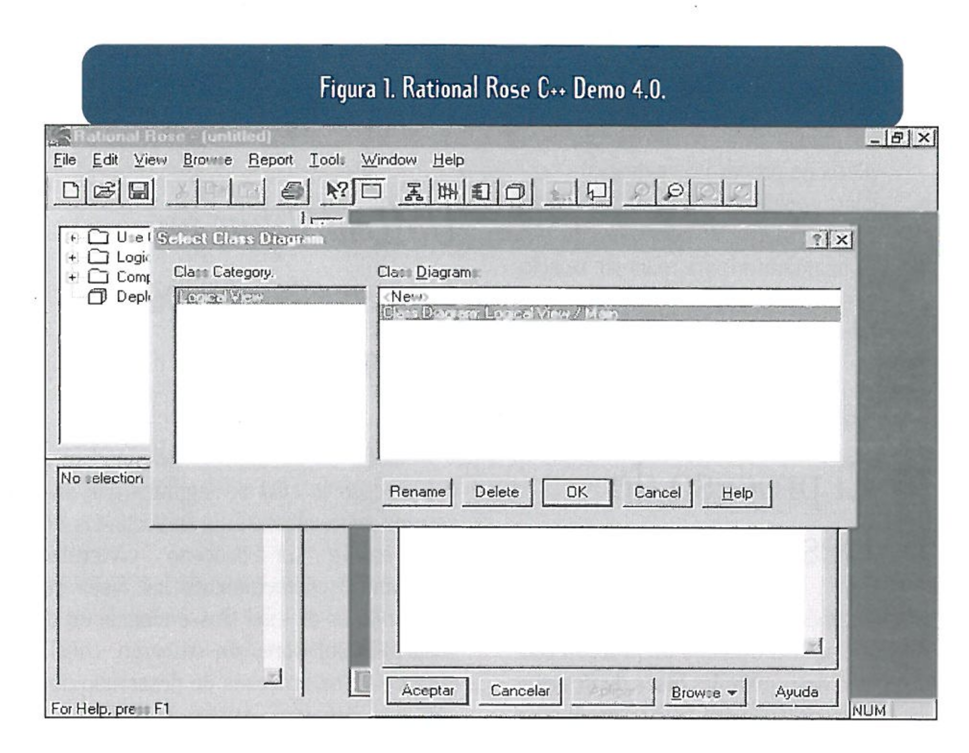


Figura 1. Rational Rose C++ Demo 4.0.

Rational Rose es la herramienta CASE que comercializan los creadores del UML y que soporta de forma completa la especificación del UML 1.0

Los creadores de UML dan mucha importancia a lo que una herramienta de desarrollo tiene que ser capaz de hacer, no en vano, al mismo tiempo que UML está siendo revisado por la OMG y por importantes empresas y desarrolladores de todo el mundo, han presentado una herramienta de diseño y análisis que utiliza el UML, conocida como *Rational Rose C++*, que en España es distribuida por la empresa Desarrollo y Macroinformación, S.A.

Fundamentándonos en la importancia que, cada vez más, tienen las herramientas de análisis y diseño en el proceso de desarrollo software, hemos revisado la herramienta Rational Rose C++ Demo 4.0 que se incluyó en el CD-ROM de esta revista en el número 34, que se basa en el método de modelado definido por UML. La Demo viene con tutoriales de infinita utilidad para el estudio de Booch, OMT y UML, que se pueden utilizar para complementar los documentos sobre la especificación UML hasta que los libros especializados sobre el tema hagan su aparición.

Rational Rose propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de dos modelos del sistema, uno lógico y otro físico. La herramienta permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema software. El modelo contiene clases, paquetes lógicos, objetos, operaciones, paquetes de componentes, módulos, procesadores, dispositivos y las relaciones que se establecen entre ellos. Cada uno de estos componentes tiene propiedades que los identifican y los caracterizan y que permiten abstraer los detalles relevantes del sistema.

La herramienta proporciona un icono gráfico para representar cada uno de estos componentes del modelo y sus relaciones, conforme a la notación que se especifica en el documento de definición del UML 1.0. Un modelo también contiene diagramas y sus especificaciones, que proporcionan la manera de visualizar y manipular los componentes del modelo y sus propiedades. Estos diagramas ilustran las múltiples vistas del modelo mientras que los iconos representan un componente que puede aparecer una, varias o ninguna vez en los diagramas del modelo.

Rational Rose permite controlar qué componentes, relaciones e iconos de propiedad aparecen en cada diagrama, utilizando las facilidades que proporciona su ventana de aplicación, donde se muestra cada diagrama en una ventana de diagrama y cada especificación en una ventana de especificación. Además de utilizar la herramienta para realizar el diseño la abundante ayuda en línea que proporciona puede resultar muy interesante como complemento para estudiar los elementos de UML.

Desarrollo Iterativo

A pesar de que el UML no identifica un proceso de desarrollo concreto, Rational Rose utiliza un proceso de desarrollo iterativo controlado (*controlled iterative process development*), donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para que éstos se hagan mínimos.

Cuando la implementación pasa todas las pruebas que se determinan en el proceso, ésta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración.

Trabajo en grupo

Rose permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo. Además utiliza ficheros para el modelo que son independientes de la plataforma, para conseguir un almacenamiento persistente de las unidades controladas y proporciona mecanismos para permitir a los ficheros que intervienen en el modelo ser copiados de un espacio de trabajo a otro.

También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades. La herramienta proporciona una ayuda muy completa que incluye guías paso a paso a través de las fases de diseño y que facilitan la comprensión de la herramienta.

Generador de Código

Una vez que se dispone de un diseño de modelo, la herramienta permite generar código utilizando la opción del Generador de Código. Este código utiliza la información que se define en el modelo UML (Rose en Rational Rose). El código que se genera viene determinado por la especificación de los componentes y las propiedades del modelo, siendo las propiedades las proporcionan la información específica que necesita el lenguaje en el que se genera el código, en este caso el C++ pero se está trabajando en generadores de código en Ada, Smallltalk y otros.

No toda la información que la herramienta puede modelar puede ser contenida en el código C++, pero la que afecta a una función particular se añade como comentarios precediendo la declaración de los miembros de la función.

Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML

Ingeniería inversa

Rational Rose proporciona mecanismos para realizar la denominada ingeniería inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño. El paquete Analizador de Rational extrae la información y la utiliza para construir un modelo que representa la estructura lógica y física de la aplicación. Rational Rose permite ver y manipular este modelo utilizando la notación UML para realizar el análisis y diseño OO. Para ello incluye un Analizador de C++ (en este caso), como un paquete ejecutable separado que se llama independientemente de la herramienta principal.

Conclusión

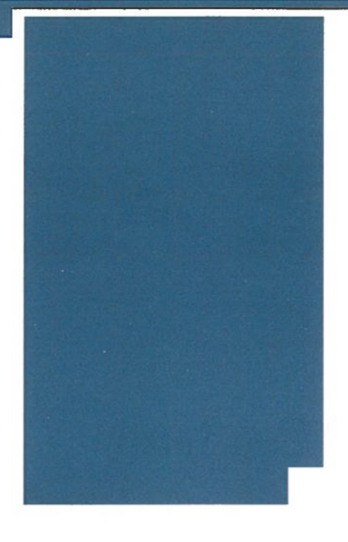
Existen infinidad de conceptos de la notación que nos dejamos en el tintero, el UML constituye un mundo por descubrir que, a medida que aparezcan en el mercado libros sobre el tema y las empresas comiencen a utilizarlo en sus proyectos, iremos conociendo. Por el momento hemos hecho una introducción a los principales elementos constituyentes del lenguaje, que estamos seguros que servirán como acicate para continuar con su estudio.

Bibliografía

Queremos agradecer a la empresa Desarrollo y Macroinformación S.A. la ayuda prestada en la realización de este artículo. Sobre UML todavía no se han publicado libros en España, en Estados Unidos existen varios libros que se encuentran en preparación (si no han sido publicados en el momento de publicarse este artículo).

Para conseguir la especificación completa del UML se puede consultar en internet: http://www.rational.com donde además se encontrará la última versión del UML.

- *C++ Manual de Referencia,* MacGraw Hill, Schildt.
- *Modelado y diseño orientados a objetos*, Prentice Hall, Rumbaugh y otros.
- UML 1.0.
- *Object Oriented Design,* Prentice Hall, Peter Coad y otros.



Correo del lector

En esta sección, los lectores de SÓLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en cualquier tema relaccionado con la programación.

Pregunta

Tengo instalada una red con un ordenador PENTIUM con Windows 95 y un 386 con Windows 3.11 usando el protocolo TCP/IP. Me interesaría utilizar el 386 como terminal del Pentium, así poder utilizar el 386 como si realmente fuese un Pentium y a la vez poder utilizar el Pentium independientemente del 386. Intento buscar en Internet algún programa que actúe como servidor Telnet para acceder desde el 386 vía Telnet al Pentium, pero no encuentro nada. ¿Se puede hacer esto?.

Respuesta

Sí, se puede hacer, pero creo que ésa no sería la mejor solución. En el caso de tener instalado Linux como software del servidor, optaríamos por configurar un daemon Telnet en el Pentium y un cliente en el 386, con lo que podrías tener acceso al sistema operativo Linux remotamente, ocupando los recursos del Pentium y no del cliente. Esto es posible porque Linux es un sistema operativo multitarea y multiusuario. Todo esto implicaría como limitación la imposibilidad de ejecutar aplicaciones gráficas, a menos que configurases un servidor X-Windows en el Pentium y tuvieras disponibilidad gráfica en el 386 además de tener un cliente de acceso a recursos X-Windows, etc. Volviendo a tu caso, no estoy al tanto de la existencia de servidores Telnet para Windows 95 y como ya te he dicho no creo que ése fuera el mejor camino. Para acceder a los recursos del servidor y aprovechar al máximo sus prestaciones, necesitas tener instalado otro protocolo adicional al TCP/IP, que en el caso de poseer las dos máquinas con Windows, nada mejor que el NetBEUI que activa la red de Microsoft. La pregunta que nos planteas lleva a confusión a mucha gente básicamente por un error de concepto. Normalmente cuando se conectan dos máquinas entre sí, lo que hacen es compartir recursos, ya sean unidades de discos, impresoras, puertos y ciertos periféricos. Cuando se ejecuta una aplicación localizada en un ordenador remoto, ésta usa los recursos del cliente como si se cargase localmente pero en una unidad diferente. La única sobrecarga del servidor se produce por un uso ajeno de su propio disco duro.

En tu caso, debemos usar un programa que *comparta* las aplicaciones, esto es, se arranquen en el servidor Pentium y envíe toda la información gráfica que se muestra en la pantalla a través de la red al cliente 386, a fin de que éste pueda tomar el control de la misma comunicándole también a través de la red, pero esta vez al servidor, las opciones que se van seleccionando.

El método explicado justifica la independencia del hardware de la estación remota o cliente con relación a la aplicación ejecutada en el servidor. Los requerimientos de la misma los proporciona el servidor y las únicas limitaciones impuestas por el cliente son aquellas que dependen del propio software usado para compartir las aplicaciones.

En el CDROM proporcionamos el **Netmeeting 2.0** que dispone de esta característica de compartir aplicaciones a través del protocolo TCP/IP, la limitación que impone es el número de colores con los que trabaja, creo que son sólo 16, deteriorando a veces la apariencia de la aplicación en el cliente si ésta requiere 256 o más. Existen otros programas además del Netmeeting que permiten realizar esto. La versión **2.0 de Connectix VideoPhone** posee una utilidad adicional para compartir aplicaciones. El precio de este último son 11.832 (IVA incl.). Ambos programas deben instalarse en los dos equipos a conectar, precisan del protocolo TCP/IP y actúan en los dos sentidos, pudiendo compartir programas de los dos ordenadores entre sí. La única objeción que veo a la solución de tu problema es la disponibilidad de dicho software para tu cliente. El programa Connectix Video Phone sí funciona en Windows 3.1x, pero precisa de un 486 DX2 y mínimo 8 Mb de RAM, aunque es posible que la característica de compartición funcione bien, puesto que estos requerimientos son los generales y esta aplicación se trata de una utilidad de videoconferencia. La versión de evaluación por 30 días de este último la puedes encontrar en el CD de la revista.

Ernesto Schmitz

Internet Explorer v4.0, DirectX 5.0 S.D.K., DJGPP y Howtos de Linux



Este mes contábamos con la llegada del nuevo navegador de Microsoft, el Internet Explorer 4.0, que ha aparecido con fuerza como anticipo a lo que será la presentación del nuevo Windows el año que viene.

Internet Explorer 4.0

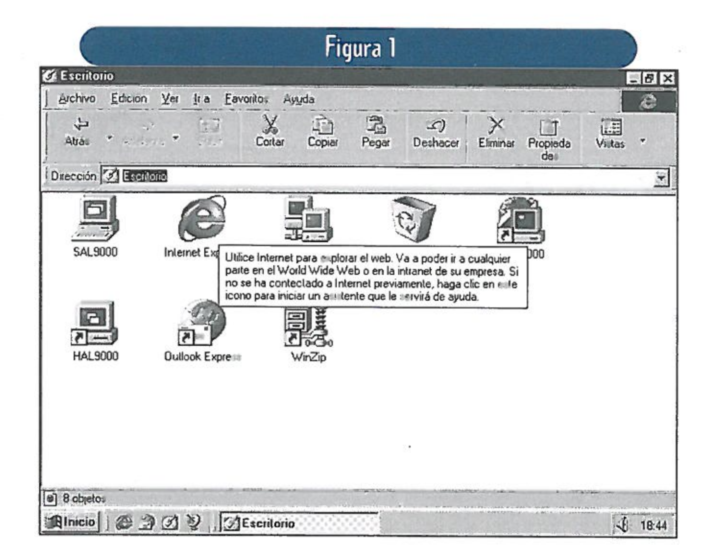
Si su máquina tiene suficientes recursos como para instalarlo, podrá comprobar que constituye, en si mismo, una remodelación a fondo de la interfaz gráfica de Windows 95; esta vez no se han conformado con añadir al sistema todos los adelantos multimedia y de vídeo conferencia existentes, como NetShow o NetMeeting, sino que además han convertido al navegador en el centro mismo del entorno fundiendo en una misma herramienta al Explorador de Windows 95 y al Explorador de la red Internet. Esto constituye la prueba de que los ordenadores personales no se conciben sin una conexión a Internet y ésta es la apuesta de Bill Gates, que en pocos años todo el mundo posea en su casa un terminal conectado como el que ahora posee un teléfono. Además de las mejoras en el GUI, el Explorer 4.0 incluye lo último en tecnología ActiveX y lo último en la red, el uso de canales.

GetRight 2.11 & VideoPhone 2.5

En este apartado, el de redes, hemos puesto a disposición del usuario dos versiones de evaluación de los programas VideoPhone y GetRight. La primera es una de las muchas, y quizá de las más conocidas, aplicaciones encargadas de facilitar a los poseedores de una mini-cámara y una tarjeta capturadora de vídeo, facilidades de vídeo conferencia (siempre que poseamos una buena conexión, por ejemplo RDSI) ya sea punto a punto, o a través de la omnipresente Internet.

La segunda es una utilidad que quizá muchos de ustedes ha venido buscando en Internet desde hace tiempo; existen herramientas FTP (protocolo de transferencia de ficheros) como CuteFTP, que incluíamos el mes pasado que son capaces, si el servidor lo permite, de recuperar los llamados *broken-downloads*, es decir, cuando se interrumpe la conexión habiendo recibido la mitad de un fichero; esto no era posible hasta ahora usando el protocolo HTTP (protocolo de transferencia de hiper-texto), y los navegadores actuales no facilitaban esta opción. Ahora sí es posible usando GetRight, que no sólo hace eso, sino que además, se da cuenta de los ficheros que hemos perdido y permite ser programado para recuperarlos sin que el usuario esté pendiente de su funcionamiento.

Figura 1



Evidentemente en este número podríamos haber incluido muchas más aplicaciones relacionadas con Internet, y por ello saldrá a la venta un número especial sobre Internet, en el que

encontraréis suculentos productos que abarquen este campo lo más ampliamente posible.

Linux

El aficionado a la informática encuentra en el sistema operativo Linux un verdadero reto para satisfacer sus deseos masoquistas; no hay nada como un entorno multitarea y multiusuario, para perder mañanas y tardes enteras configurándolo, recompilando el Kernel, preparando los servidores de X-Window y volviéndose loco con cientos de pequeños detalles que tener en cuenta. Pero como no siempre tenemos el tiempo ni la paciencia suficientes para descubrir cómo hacer cada una de las cosas que nos permite Linux, existen una serie de ficheros, normalmente en formato ASCII, explicando paso a paso cómo hacer todo, o casi todo. Esperamos que los HowTos que os proporcionamos os sirvan de utilidad.

Para los buenos aficionados al Linux, que sabemos que son muchos, hemos incluido la versión definitiva del Netscape v3.0 y además la última distribución del *Java Development Kit*, la 1.1.3, para el único sistema operativo "free".

DJGPP

DJGPP incluye un compilador y varias herramientas de desarrollo para un entorno en modo protegido (*protected-mode*) de programación de CPUs Intel de 32 bits ejecutando sistemas operativos MS-DOS y compatibles. No necesita extensores del MS-DOS para ejecutarse, sólo un servidor DPMI.. DJGPP incluye uno de libre distribución de 32 bits que permite programar en un espacio de direcciones que va desde 4Gb en direcciones planas, hasta 256Mb en memoria virtual, con un compilador que produce código en modo protegido de 32 bits y una suite de herramientas de desarrollo GNU portadas a MS-DOS. Esto nos proporciona un entorno de trabajo que favorece específicamente la portabilidad de nuestros programas a entornos Unix, aunque también es bueno para la creación de nuevo código. Está especialmente indicado para los programadores de juegos y programas que exijan un gran conocimiento de la máquina, sus dispositivos y drivers.

DirectX 5.0

También incluimos el último conjunto de drivers de Microsoft que implementan nuevas y mejoradas capacidades multimedia para su uso por vídeo juegos de última generación. En esta nueva revisión se ha implementado una gran conexión con Internet Explorer 4.0, de manera que páginas HTML visualizadas con este navegador pueden tener acceso a los mismos. Junto con los drivers hemos incluido el SDK que ayudará a los programadores a desarrollar nuevos y excitantes contenidos multimedia. Pero sólo para Windows 95, los usuarios de Windows NT 4.0, deberán instalar el Service Pack 3 para incluir DirectX3 en sus sistemas.

Ayudando al Programador

A petición de nuestros lectores incluimos varias herramientas con diferentes propósitos:

• Irfan Viewer v2.5 Freeware, un visualizador/Convertidor de ficheros gráficos Freeware. Entre los formatos soportados destacan: JPEG, GIF, BMP, DIB, RLE, PCX, PNG, TIFF, TGA, RAS/SUN, ICO, PBM, PGM, PPM, IFF/LBM y Photo-CD.

• Programmer's File Editor, un editor de ficheros de texto para Windows 95 y NT destinado a los programadores. Permite la lectura-escritura de ficheros de gran tamaño con un mínimo consumo de memoria. Posee muchas opciones configurables por el usuario así como capacidad para trabajar con varios ficheros a la vez.

OpenGL

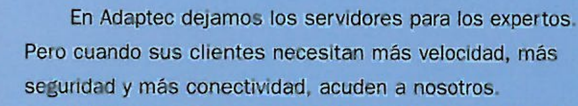
OpenGL es la abreviatura de *Open Graphics Library* y consiste en la estandarización de lo que en su día fueron la librerías GL de Silicon Graphics. OpenGL posee cientos de funciones con las que puedes realizar *renders* (rellenados) en dos y tres dimensiones usando el algoritmo Z-buffer. Algunas implementaciones de OpenGl como las que funcionan sobre los PCs operan completamente en software mientras que otras se aprovechan del hardware especializado disponible para algunas estaciones de trabajo. Hay una implementación de libre distribución de estas librerías con "no-licencia" GNU llamada Mesa, y una versión para la mayoría de las versiones de Unix (incluyendo Linux) y las versiones de 32-bits de Microsoft Windows como 95 y NT. También hay una versión para Macintosh pero es un poco complicada de instalar. Aquí incluimos las librerías estándar que un programa que utiliza OpenGL solicitaría al sistema, esperamos que os sean de utilidad, está en estudio que en próximos CD-ROMs incluyamos todo el entorno de desarrollo necesario para programar bajo las restricciones de OpenGL.

Help Wanted

Como siempre instamos al lector que nos solicite aquellos productos que puedan serle de utilidad siempre que éstos sean de libre distribución o tengan una licencia restringida para un periodo de prueba. Sabemos que esto evita penosas cuentas de teléfono y esperas interminables para aquellos que están conectados a la red y facilita el acceso a recursos, que de otra manera serían inaccesibles, de aquellos que no están conectados. De esta manera nos mantenemos al día con las necesidades del mercado y el CD-ROM llevará un contenido acorde con los gustos de los fieles lectores de Sólo Programadores.

Ya conocéis nuestra dirección de correo electrónico: solop@towercom.es. Utilizadla tanto como deseéis.

No hacemos servidores, simplemente los mejoramos



En Adaptec dejamos los servidores para los expertos. Pero cuando sus clientes necesitan más velocidad, más seguridad y más conectividad, acuden a nosotros.

SCSI - Los adaptadores SCSI de Adaptec le permiten conectar hasta 15 periféricos a su servidor y son capaces de doblar la velocidad a la que puede mover los archivos por el sistema.

RAID - La serie de adaptadores de matriz AAA-130 de Adaptec hace de RAID una opción viable para servidores con Windows NT y NetWare. Considerado como el concepto más fiable de almacenamiento de datos, RAID divide los datos y los coloca en varias unidades, con lo que le proporciona un acceso más rápido y seguro.

FAST ETHERNET - Si desea conectar su servidor a una red Fast Ethernet, el adaptador de 4 puertos Quartet de Adaptec cuadruplica su número de conexiones y sostiene velocidades de hasta 100 Mb/s. Quartet también es fácil de administrar con la suite de software Duralink.

FIBRE CHANNEL - Para almacenamiento externo y grupos de servidores, los controladores Fibre Channel de Adaptec aseguran un rendimiento inmejorable. Fibre Channel es la última tecnología para subsistemas de almacenamiento en servidor y proporciona la friolera de 100 Mb/s en distancias de hasta 10 Km para hasta 126 dispositivos.

Si su servidor es simplemente estándar, su rendimiento también lo será. Póngase en contacto con Adaptec, el líder mundial en tecnología de transferencia de datos, y juntos haremos que su servidor sea superior.

http://www.adaptec.com.

We move the information that moves your world.

